

# Notices

---

## Authorship

The NET/ROM software was written by Ronald E. Raikes, WA8DED. Concept, design, and documentation by Michael D. Busch, W6LXU.

## Copyright and Trademark

*This document © 1987 Software 2000, Inc.  
NET/ROM software © 1987 Software 2000, Inc.  
NET/ROM™ is a trademark of Software 2000, Inc.*

The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of the agreement. *It is against the law to make reproductions of NET/ROM software on ROM, magnetic tape, disk, or any other medium for any purpose whatsoever.*

## Warranty

Software 2000, Inc., warrants that it has the right to license the NET/ROM software, and agrees to replace defective copies of NET/ROM within 90 days of purchase provided the original ROM is returned postpaid. Software 2000, Inc., makes no other warranties, either express or implied, regarding NET/ROM software, its merchantability or its fitness for any particular purpose.

## Disclaimer

Information in this document is subject to change without notice, and is not a binding commitment on the part of Software 2000, Inc.

## For more information...

**Software 2000, Inc.  
1127 Hetrick Avenue  
Arroyo Grande, California 93420  
U.S.A.**

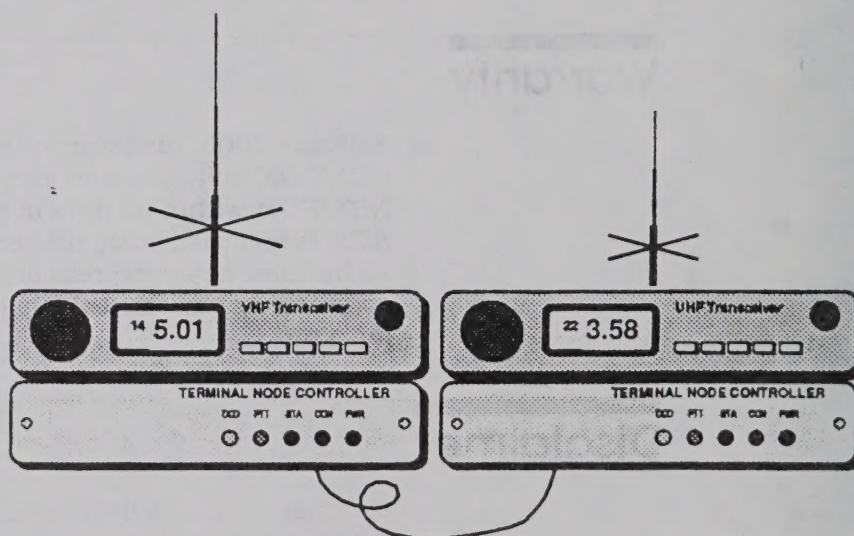
Telephone: (805) 489-1977  
Telex: 658541 (sftwre argd ud)  
CompuServe: 76337,727

---

# **NET/ROM** <sup>TM</sup>

**Version 1.3 Documentation**  
**September, 1987**

---



---

Software 2000, Inc.  
1127 Hetrick Avenue  
Arroyo Grande, California 93420  
U.S.A.

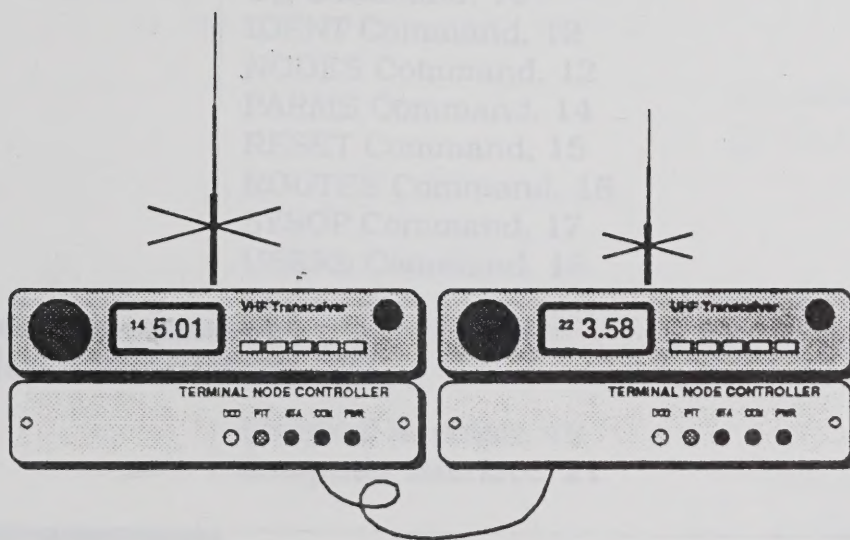
---

---

# ***NET/ROM***

***Version 1.3 Documentation  
September, 1987***

---



---

Software 2000, Inc.  
1127 Hetrick Avenue  
Arroyo Grande, California 93420  
U.S.A.

---



# **Contents**

---

## **Introduction, 1**

## **A Guided Tour, 5**

- Connecting to a Nearby Station, 5
- Connecting to a Distant Station, 6
- Connecting to a NET/ROM Host, 7
- Calling CQ and Answering CQs, 8

## **Commands, 9**

- CONNECT Command, 9
- CQ Command, 10
- IDENT Command, 12
- NODES Command, 12
- PARMS Command, 14
- RESET Command, 15
- ROUTES Command, 16
- SYSOP Command, 17
- USERS Command, 18

## **Host Interface, 19**

- Terminal Interface, 19
- Computer Interface, 21

## **Installation, 27**

- TNC-2 Hardware Setup, 27
- PK-87/PK-90 Hardware Setup, 30
- Software Setup, 30
- Dual- or Multi-Channel Installations, 32

---

---

## Theory of Operation, 33

Making Connections, 33  
Digipeating vs. Store-and-Forward, 37  
Structure of the Firmware, 39  
Internal Information Flow, 42  
Automatic Routing Mechanism, 45  
Some Other Facilities, 48

---

## Node Notes, 51

Notes for Users, 51  
Notes for Control Operators, 54

---

## Protocols, 59

Structure of Inter-Node HDLC Frames, 59  
Transport Layer (End-to-End) Protocol, 60  
RS232 Interconnect Protocol, 62

---

## Routing, 63

Routing Table Structure, 63  
Routing Algorithm, 64  
Automatic Routing Table Updates, 65

---

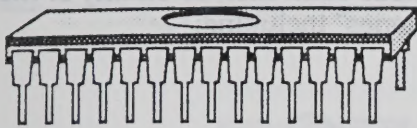
## Parameters, 67

---

## Index, 73

---

# Introduction



**N**ET/ROM is a firmware replacement which transforms an ordinary TNC-2 or PK-87 terminal node controller into a state-of-the-art network node controller. The standard version of NET/ROM is designed for use on hilltops and other wide-coverage digipeater sites. A special version called "Host NET/ROM" is designed to interface with bulletin board systems and similar computer-based network servers. NET/ROM is not intended for use at end-user stations.

A NET/ROM node provides the normal functions of an ordinary AX.25 digipeater, plus a set of sophisticated higher-level networking capabilities. A user may connect to a nearby NET/ROM node; display a directory of other known network nodes; establish a transport-layer circuit to a distant node; and call CQ or connect to another station in the vicinity of the distant node.

Compared with conventional multi-hop digipeating, NET/ROM's store-and-forward packet switching can provide an order-of-magnitude improvement in reliability and throughput, especially over long paths. Routing from the local node to the distant node is handled automatically, and includes alternate routing to circumvent network outages.

NET/ROM supports cross-frequency and cross-band networking without the need for exotic multi-port digipeater hardware. A dual-channel node, for example, is implemented simply by installing NET/ROM in a pair of TNCs and connecting them together with an RS232 cable. A three- or four-channel node can be created by connecting multiple TNCs together, using a simple diode-matrix coupler.

NET/ROM uses the standard AX.25 link protocol for both node-to-node and node-to-user links at ISO layer two. Higher-level NET/ROM protocols provide a "pure datagram" network service at layer three, plus a "virtual circuit" transport service at layer four.

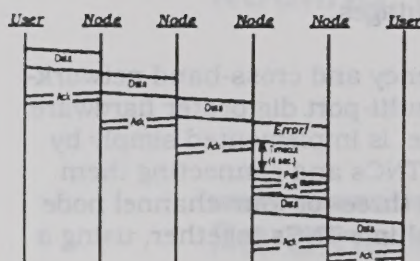
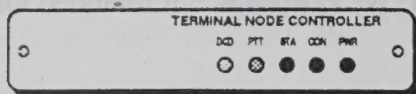
## Highlights

### Runs on ordinary TNCs

NET/ROM provides state-of-the-art networking capabilities without the need for special hardware. NET/ROM runs on a standard TAPR TNC-2 terminal node controller, or on any of the commercially-available TNC-2 "clones" including:

- AEA PK-80
- GLB TNC2A
- MFJ 1270 and 1274
- Pac-Comm TNC-200
- California Digital TNC-2
- Fuji Digital Systems DA-12F and DA-12M
- AIWA APX-25
- TASC0 TNC-20

A version of NET/ROM is also available for the AEA PK-87 and PK-90 TNCs. NET/ROM is distributed in the form of a 27C256 EPROM which simply plugs into the ROM socket of the TNC in place of the manufacturer-furnished firmware.

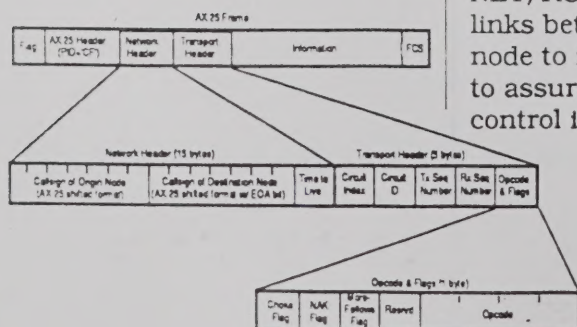


### Store-and-forward packet switching

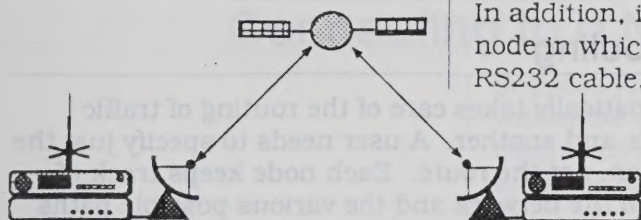
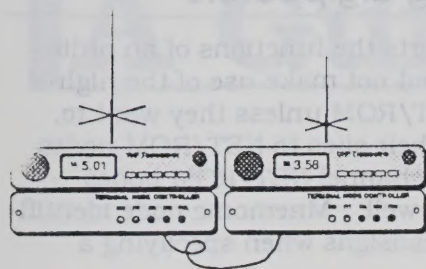
Instead of multi-hop digipeating, NET/ROM provides true store-and-forward packet switching. The result is significant improvement in throughput and reliability. On long paths (three or more digipeaters), dramatic improvements are possible. For instance, the calculated speed improvement for a path involving five "hops" (four digipeaters) assuming typical 1200-baud VHF network conditions is about 800%!

### Multi-layer protocols

NET/ROM uses the standard AX.25v2 packet radio protocol for links between neighboring nodes, as well as for links from each node to its local users. Normal AX.25v2 error handling is used to assure error-free transmission, and normal AX.25v2 flow control is used to manage network congestion.



In addition, NET/ROM incorporates a transport-layer "sliding window protocol" to provide end-to-end error and flow control for each virtual circuit. End-to-end error control protects against lost, duplicate, or out-of-sequence frames resulting from node failures and dynamic routing changes. End-to-end flow control protects the network against disproportionate loading by any one circuit.



## Multi-channel without special hardware

NET/ROM supports multi-channel operation without the need for exotic multi-port digipeater hardware. A dual-channel node, for example, consists simply of a pair of standard TNCs (with NET/ROM in each) connected together with a simple RS232 cable. Configuring a NET/ROM node for three or four channels is almost as easy—a TNC is used for each frequency, and the multiple TNCs are interconnected via their RS232 ports using a simple diode-matrix coupler.

In addition, it is possible to configure a dual-channel NET/ROM node in which the two TNCs are not co-located. Instead of an RS232 cable, the TNC interconnect can employ a dedicated telephone line or even a satellite link. Because NET/ROM uses an asynchronous variant of AX.25 over the interconnect, it is not necessary that it be an error-free connection. This opens up fascinating possibilities, such as the experimental satellite-linked dual-channel node accessible from both Baltimore and San Francisco—affectionately known as “the wormhole”!

## High-level “CQ” facility

NET/ROM allows a user to broadcast a CQ from a local or distant node, and enables other stations to reply to the CQ using the high-level facilities of the network. A user's CQ request remains active for up to 15 minutes, during which time it appears in the node's user directory. Another user that connects to the node can display this directory, learn of any stations with active CQ requests, and reply to one of them if he so desires.

## Deferred-disconnect

When two stations are connected to one another via NET/ROM and one of the stations disconnects, NET/ROM automatically maintains its connection to the other station until all in-transit information frames have been successfully delivered to that station. NET/ROM disconnects only after all in-transit information has been delivered, or after 15 minutes has elapsed without any “forward progress” in delivering such information.

## Mnemonic node identifiers

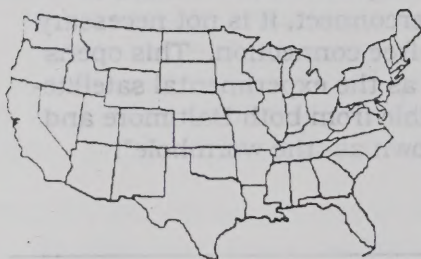
Each NET/ROM node is identified in two ways: by a valid amateur callsign permanently encoded into each copy of NET/ROM, and by an arbitrary mnemonic node identifier established by the node's control operator. Identifiers may be up to six characters long—three-letter city designators used in aviation (LAX, SFO, DCA, ATL, JAX, etc.) are a possible choice. Mnemonic node identifiers appear in node station identification beacons, and are passed to other nodes during hourly automatic routing broadcasts. The node identifier may be used in lieu of the node callsign in most contexts.

**CQ CQ CQ CQ CQ CQ**

**disc**

**LAX**

# via



Signature FF hex	Mnemonic ident of sending node (6 bytes)	Call sign of destination node (7 bytes)	Mnemonic ident of destination node (6 bytes)	Call sign of best quality neighbor (7 bytes)	Best quality value	(more destinations)
Repeat for each known destination (up to 11 per UK frame)						



## Compatible with existing digipeaters

Each NET/ROM node also supports the functions of an ordinary AX.25 digipeater. Users need not make use of the high-level networking functions of NET/ROM unless they want to. Digipeater owners can upgrade their sites to NET/ROM nodes without disrupting users. Multi-channel NET/ROM nodes provide multi-port digipeating as well. Mnemonic node identifiers may be used in lieu of node callsigns when specifying a digipeated path.

## Automatic routing

NET/ROM automatically takes care of the routing of traffic between one node and another. A user needs to specify just the desired destination, not the route. Each node keeps track of the other nodes in the network and the various possible paths that may be used to reach them. If a node or path becomes unusable due to equipment failure or poor propagation, NET/ROM automatically changes to an alternate route (if available) to circumvent the outage. Conversely, when a new node is placed on-line, other nodes automatically incorporate the new node into the network routing structure. Such routing changes are handled dynamically, without disrupting user connections in progress.

NET/ROM supports three methods of updating its routing information: local, remote, and automatic. Initial routing information may be entered manually by an on-site operator using a local terminal. Routing changes may be made remotely

by a control operator over an ordinary packet radio connection—a randomized verification algorithm effectively prevents changes by

unauthorized operators. In addition, NET/ROM nodes broadcast routing information to each other on an hourly basis, thereby enabling the network to incorporate new nodes and to bypass outages in real-time without manual intervention.

## “User-friendly”

Despite its internal sophistication and advanced networking capabilities, NET/ROM is exceptionally easy to use. A novice user needs to learn only one command, CONNECT, to establish crosslinks to other nodes or downlinks to other user stations. More sophisticated users may wish to use the CQ command to solicit contacts, the NODES command to get a directory of other network nodes, and the USERS command to find out who else is using the node.

Control operators can use SYSOP to validate their control operator privileges, NODES and ROUTES to make manual changes to the routing table, IDENT to establish a mnemonic node identifier, PARMS to set or display various node parameters, and RESET to warm-start the NET/ROM firmware.

# A Guided Tour

---

If you operate a conventional AX.25 packet radio station, you shouldn't have any difficulty at all learning to use NET/ROM. This chapter will show you everything you need to know to make effective use of the NET/ROM network.

---

## Connecting to a Nearby Station

Suppose you want to connect to another station that is located within the coverage area of your local NET/ROM node. All you need to do is connect your TNC to the local node, and then ask the node to connect you to the other station. Here's how...

### First, uplink...

The first step in using the NET/ROM network is always the same: establish an "uplink" to your local node. Use the normal connect command provided by your TNC. You can use either the node's callsign or its mnemonic identifier. For instance, let's assume that you are located in Las Vegas, Nevada. The callsign of your local NET/ROM node is K7WS-1, and its identifier is LAS. To uplink, connect either to K7WS-1 or to LAS:

```
^C
cmd: CONNECT LAS
*** 'CONNECTED to LAS
```

Note: your keyboard input is in *bold italics* for clarity.

### Then, downlink...

Now you are talking to the Las Vegas node. Suppose you want to connect to WB7Z, who is also located within the coverage area of the Las Vegas node. All you have to do is to ask the node to make the connection for you:

```
CONNECT WB7Z
LAS:K7WS-1) Connected to WB7Z
Hello Randy, this is Mike. How's everything?
Fine, Mike. Hot enough for you today?
etc.
```

In response to your CONNECT command, the node has established a "downlink" to WB7Z for you. When you receive the "Connected to..." message, it means that the node has successfully connected you to the other station. At this point, you can ignore the node and carry on just as if you were directly connected to the other station.

## When done, disconnect...

When you are done conversing with WB7Z, you can simply disconnect your TNC as usual:

```
^C
cmd: DISC
*** DISCONNECTED
```

## If something goes wrong...

Naturally, things don't always go quite this smoothly. For example, if WB7Z's station is busy, you might see:

```
CONNECT WB7Z
LAS:K7WS-1} Busy from WB7Z
```

Or if it simply isn't on the air:

```
CONNECT WB7Z
LAS:K7WS-1} Failure with WB7Z
```

In any case, the node will let you know if something goes wrong. You can try again, do something else, or simply disconnect.

# Connecting to a Distant Station

OK, now suppose you want to connect to a distant station served by a different node. This time, the process involves three steps: *uplink* to your local node, *crosslink* to the distant node, and then *downlink* to the other station. Here goes...

## First, uplink...

As before, the first step is to uplink to your local node:

```
^C
cmd: CONNECT LAS
*** CONNECTED to LAS
```

Once again, you are now talking to the Las Vegas node. At this point, it might be a good idea to find out what other nodes are accessible from Las Vegas (although this step is not required). You can ask for a directory of other nodes with the **NODES** command:

```
NODES
LAS:K7WS-1} Nodes:
ABJO:KA7LEG-1  ANGEL:K7WS-2  BGBEAR:AA6TN-1
BLU:KD7UD-1    CDC:WA7GTU-1  COE:KK7X-4
COP:K7MM-5     ELY:WB7WTS-1  FORD:KD7YG-1
FRSCO:WA7GTU-2 KEM:KD7YK-3   NLAS:K7WS-3
OREM:KD7YK-1   RENO:AK7B-14  RWL:W7KF-1
SLC:K7EA-1     SLC2:K7EA-2   SNOW:KD7YK-2
WMS:K7WS-4
```

## Next, crosslink...

This time, suppose you want to connect to W7LN in Salt Lake City, Utah. The NODES display has confirmed that the Salt Lake City nodes SLC:K7EA-1 and SLC2:K7EA-2 are accessible from Las Vegas. So the next step is to connect to one of these nodes:

```
CONNECT SLC
LAS:K7WS-1} Connected to SLC:K7EA-1
```

In response to your CONNECT command, the node has established a "crosslink" for you to the SLC:K7EA-1 node. The "Connected to..." message means that you are now talking to the Salt Lake City node.

## Finally, downlink...

Your friend W7LN is in the local coverage area of the SLC node, so all that's left is to ask the node to downlink to him, just as before:

```
CONNECT W7LN
SLC:K7EA-1} Connected to W7LN
Hello Tom! The network is working great today.
Long time no hear, Mike.
etc.
```

Once again, if anything goes wrong, the node will give you a "Busy from..." or "Failure with..." message to let you know.

---

## Connecting to a NET/ROM Host

Some NET/ROM nodes have a bulletin-board system or other computer-driven server attached to them. To access such a "host" system, a slightly different procedure is involved.

First, connect to the node as before, by means of an uplink and (if the node is distant) a crosslink. Then, to access the host system, issue a CONNECT command *with no callsign*. Here's how it looks:

```
^C
cmd: CONNECT LAS
*** CONNECTED to LAS
CONNECT LGU
LAS:K7WS-1} Connected to LGU:WA7MBL-1
CONNECT
LGU:WA7MBL-1} Connected to LGU:WA7MBL-1
This is the WA7MBL-1 BBS in Logan, Utah
Hello, Mike.
Command >
etc.
```

---

## Calling CQ and Answering CQs

Finally, if you want to make contact with *any* station in an area (rather than with a *particular* station), NET/ROM allows you to connect to a node (local or distant) and either call "CQ" or reply to another station's "CQ".

First, connect to the desired node. Then, to check if any other stations are calling CQ at the node, issue a USERS command:

```
^C
cmd: CONNECT LAS
*** CONNECTED to LAS
CONNECT BGBEAR
LAS:K7WS-1) Connected to BGBEAR:AA6TN-1
USERS
BGBEAR:AA6TN-1) NET/ROM 1.3 (707)
Uplink (WB6UUT)
Uplink (WA8DED) <--> CQ (WA8DED-15)
Circuit (LAS:K7WS-1 W6IXU)
```

The USERS display shows that WA8DED is calling CQ at the BGBEAR node. To answer a CQ, simply issue a CONNECT command with the callsign specified in the CQ(...) portion of the USERS display:

```
CONNECT WA8DED-15
BGBEAR:AA6TN-1) Connected to WA8DED
This is Ron in Downey. Thanks for the call.
etc.
```

Alternatively, you can call CQ yourself by issuing a CQ command to the node:

```
CQ from "Mike" in Las Vegas, Nevada
```

Once you issue a CQ command, you should wait for someone to reply. Your CQ remains active in the node's USERS display for fifteen minutes after you issue it, so there's no point in issuing additional CQs. When someone answers your CQ, the node will send you a "Connected to..." message:

```
CQ from "Mike" in Las Vegas, Nevada
BGBEAR:AA6TN-1) Connected to NK6K
Hi, this is Harold in Redondo Beach, Calif.
etc.
```

After fifteen minutes, if there has been no reply to your CQ and you haven't issued any further commands, your TNC will disconnect.

*Note: The CQ command was introduced in NET/ROM version 1.3. On a node using an earlier version, you will get the message "Invalid command".*

# Commands

---

**N**ET/ROM supports nine commands: CONNECT, CQ, IDENT, NODES, PARMS, RESET, ROUTES, SYSOP, and USERS. For any of them, the entire command verb ("CONNECT") or just a fragment ("CONN" or "CO" or "C") is allowed. ("C" is taken to mean CONNECT, not CQ. "R" is taken to mean ROUTES, not RESET.) Any command parameters must be separated from the verb and each other by one or more spaces. The maximum command length is 80 characters. Commands must end with a carriage-return.

---

## CONNECT Command

The CONNECT command is used to request a circuit to another node, a downlink to another user, or a connection to the node's host terminal.

To request a circuit to another node, use:

`CONNECT node`

where *node* must be the callsign or mnemonic identifier of another node that is "known" to this node. (Use the NODES command to obtain a list of all known node callsigns and identifiers.) For example:

`CONNECT WB7BNI-1`  
LAS:K7WS-1} Failure with PHX:WB7BNI-1

`CONNECT SLC`  
LAS:K7WS-1} Connected to SLC:K7EA-1

To request a downlink to another user, use:

`CONNECT usercall [[VIA] digicall,...,digicall]`

where *usercall* is the callsign of the user station, and *digicall* is the callsign (or alias) of a digipeater. If digipeaters are used, the use of "VIA" is optional ("VI" or "V" are also acceptable). Digipeaters may be separated by either spaces or commas. For example:

`CONNECT WM6P via N6GPP-1`  
BGBEAR:AA6TN-1} Busy from WM6P

`CONNECT W0RLI`  
SFO:W6AMT} Connected to W0RLI

To request a connection to the node's host terminal or host computer, the command is:

**CONNECT**

with no parameters. For example:

**CONNECT**

SBA:W6AMT-2) Connected to SBA:W6AMT-2

In all cases, a successful connection is announced by the message "Connected to...". "Failure with..." indicates that the specified node or user did not respond after a number of attempts. "Busy from..." indicates that the node or user responded but refused the connection request.

Other possible error messages are "Node busy", "Circuit table full", "Link table full", and "Host table full". These messages indicate a lack of resources in the node—the user should disconnect and try again later.

An in-process CONNECT command is immediately aborted if another command line (valid or not) is entered before the requested connection is established.

---

## CQ Command

The CQ command is used to broadcast a short text message from a node, and to enable other user stations that receive the broadcast to connect to the station that originated the broadcast. The command is:

**CQ [textmessage]**

where *textmessage* is optional and can be any string up to 77 characters long (blanks and punctuation are allowed). Note that the CQ command cannot be abbreviated, since "C" is interpreted as a CONNECT command.

In response to a CQ command, the node broadcasts the specified message in "unproto" mode, using the callsign of the originating user (with a translated SSID) as the source and "CQ" as the destination. The broadcast is made in the form of an AX.25 UI-frame with a PID of 'F0' hex. For example, if user station W1XYZ connects to a node and issues the command:

**CQ "Mike" in Las Vegas, Nevada**

the node transmits a broadcast that would be monitored by local users as:

W1XYZ-15>CQ: "Mike" in Las Vegas, Nevada

After making the broadcast in response to the CQ command, the node "arms" a mechanism to permit other stations to reply to the CQ. A station wishing to reply may do so simply by connecting his TNC to the originating callsign shown in the broadcast (W1XYZ-15 in the example above). A CQ command remains "armed" to accept replies for 15 minutes (see PARMS parameter 15), or until the originating user issues another command or disconnects from the node.

Any station attached to a node in command mode may determine if there are any other stations awaiting a reply to a CQ by issuing a USERS command. An "armed" CQ channel appears in the USERS display as:

*{Circuit, Host or Uplink} <~~> CQ(usercall)*

The station may reply to such a pending CQ by issuing a CONNECT to the user callsign specified in the CQ(...) portion of the USERS display—it is not necessary for the station to disconnect from the node and reconnect. For example:

```
^C
cmd: C BWI
*** Connected to BWI ***
USERS
BWI:W3IWI-5) NET/ROM 1.3 (701)
Uplink(K1HTV-1) <~~> CQ(K1HTV-14)
Circuit(LAS:K7WS-1 W1XYZ) <~~> CQ(W1XYZ-15)
Uplink(N4HY)
CONNECT W1XYZ-15
BWI:W3IWI-5) Connected to W1XYZ
Hi, Dr. Bob! Thanks for answering my CQ.
etc.
```

Users of the CQ command are cautioned to be patient in awaiting a response. Your CQ will remain "armed" for 15 minutes, and will be visible to any user who issues a USERS command during that time. Consequently, there's no point in issuing additional CQs—give other stations a chance to reply to your first one!

*NOTE: The CQ command was introduced in NET/ROM version 1.3. On a node using an earlier version, you will get the message "Invalid command".*

---

## IDENT Command

The IDENT command allows an authorized control operator to set or change the node's mnemonic identifier. The command is:

**IDENT *identifier***

where *identifier* is a string up to six characters long, or "\*" if the node is to have no identifier at all. For example:

```
IDENT LAS
LAS:K7WS-1} LAS
```

```
IDENT *
K7WS-1}
```

Node identifiers should normally be composed of letters and digits only. Non-printing characters and punctuation marks are invalid (with one exception discussed in the next paragraph). Lower-case letters are converted to upper-case. (In addition, the amateur version of NET/ROM will not accept a node identifier that "looks like" a valid amateur callsign—a string of letters and digits looks like a callsign if it is four to six characters long, has either one or two digits, and the rightmost digit is neither the first nor the last character in the string.)

A node identifier may use the character "#" in its first character position. This causes the node to be suppressed from the NODES displays at other nodes. (NOTE: The use of other punctuation characters in node identifiers is reserved for possible future extensions of NET/ROM.)

Before using the IDENT command to change the node identifier remotely, you must validate your credentials as a control operator by using the SYSOP command...otherwise, the node identifier is left unchanged.

---

## NODES Command

The NODES command is used to display or modify the destination list of the node's routing table.

To display a list of other known destination nodes, use NODES without any parameters:

**NODES**

LAS:K7WS-1} Nodes:

ABJO:KA7LEG-1	ANGEL:K7WS-2	BGBEAR:AA6TN-1
BLU:KD7UD-1	CDC:WA7GTU-1	COE:KK7X-4
COP:K7MM-5	ELY:WB7WTS-1	FORD:KD7YG-1
FRSCO:WA7GTU-2	KEM:KD7YK-3	NLAS:K7WS-3
OREM:KD7YK-1	RENO:AK7B-14	RWL:W7KF-1
SLC:K7EA-1	SLC2:K7EA-2	SNOW:KD7YK-2
WMS:K7WS-4		

The normal NODES display includes all known nodes in the routing table except "hidden nodes" with mnemonic identifiers that start with the character "#". To display of all nodes *including* such hidden nodes, use the command "NODES \*":

**NODES \***

LAS:K7WS-1) Nodes:

#BGBR6:AA6TN-6	#LAS:K7WS-11	#SLC2:K7EA-12
ABJO:KA7LEG-1	ANGEL:K7WS-2	BGBEAR:AA6TN-1
BLU:KD7UD-1	CDC:WA7GTU-1	COE:KK7X-4
COP:K7MM-5	ELY:WB7WTS-1	FORD:KD7YG-1
FRSCO:WA7GTU-2	KEM:KD7YK-3	NLAS:K7WS-3
OREM:KD7YK-1	RENO:AK7B-14	RWL:W7KF-1
SLC:K7EA-1	SLC2:K7EA-2	SNOW:KD7YK-2
WMS:K7WS-4		

To display specific routing information for a particular node, use NODES followed by the callsign or identifier of the node in question:

**NODES SLC**

LAS:K7WS-1) Routes to SLC:K7EA-1

```
> 126 6 1 K7WS-11
   108 5 0 WA7GTU-1
   0 6 0 AA6TN-1
```

This command shows up to three routes to the specified node. For each route, the following items are displayed:

- the symbol ">" if the route is in use
- quality of the route (255 is best, 0 is worst)
- obsolescence count (0 denotes a locked entry)
- port number (0=HDLC port, 1=RS232 port).
- path to neighboring node (callsign + any digis)

The NODES command also supports manual updates to the routing table, but this capability is available only to a control operator who has previously validated his credentials during this connection by successfully executing the SYSOP command. To add or delete routing table entries, the commands are:

```
NODES nodecall + ident quality count port neighbor [digicall...]  
NODES nodecall - ident quality count port neighbor [digicall...]
```

The "+" version adds a new entry to the routing table; the callsign *nodecall* is added to the list of known nodes if it isn't already there. The *ident* parameter is a mnemonic identifier up to six characters long, or "\*" if the node has no identifier. The *quality* parameter is a decimal integer in the range 255 (best) to 0 (worst) that defines the quality of the best-case route to the destination node via the specified neighbor. The *count* parameter is the initial obsolescence count—zero denotes a permanent, unchangeable route entry. The *port* parameter is given as either 0 (HDLC port) or 1 (RS232 port). The *neighbor* and *digicall* parameters give a path to an adjacent node that is enroute to the destination (two digipeaters max).

The "-" version searches the routing table for an entry that exactly matches the given *nodecall*, *port*, *neighbor* and *digicall* parameters, and deletes it if such an entry is found; deleting the last route for a particular nodecall also removes it from the list of known nodes. In case of an invalid request (or a valid request by an unvalidated user), these commands are quietly ignored, not diagnosed. Examples:

```

NODES K7EA-1 + SLC 115 6 0 AA6TN-1
LAS:K7WS-1} Routes to SLC:K7EA-1
> 126 6 1 K7WS-11
    115 6 0 AA6TN-1
    108 5 0 WA7GTU-1

```

```

NODES K7EA-1 - SLC 108 5 0 WA7GTU-1
LAS:K7WS-1} Routes to SLC:K7EA-1
> 126 6 1 K7WS-11
    115 6 0 AA6TN-1

```

## PARMS Command

The PARMS command can be used to display or change various numeric parameters that affect operation of the node. There are 26 node parameters that can be set by the PARMS command:

<u>No.</u>	<u>Description of Parameter</u>	<u>Default</u>	<u>Min</u>	<u>Max</u>
1.	Max destination list entries	50	1	400
2.	Worst quality for auto-updates (0 disables)	1	0	255
3.	Channel 0 (HDLC) quality	192	0	255
4.	Channel 1 (RS232) quality	255	0	255
5.	Obsolescence count initializer	6	0	255
6.	Obsolescence count min. to be broadcast	5	1	255
7.	Auto-update broadcast interval (sec)	3600	0	65535
8.	Network "time-to-live" initializer	64	0	255
9.	Transport timeout (seconds)	60	5	600
10.	Transport maximum tries	3	2	127
11.	Transport acknowledge delay (seconds)	3	1	60
12.	Transport busy delay (seconds)	180	1	1000
13.	Transport requested window size (frames)	4	1	127
14.	Congestion control threshold (frames)	4	1	127
15.	No-activity timeout(seconds)	900	0	65535
16.	Keyup P-persistence threshold (P/256)	64	0	255
17.	Keyup slot time (10ms increments)	10	0	127
18.	Link T1 timeout "FRACK" (seconds)	4	1	15
19.	Link tx window size "MAXFRAME" (frames)	7	1	7
20.	Link maximum tries (0=try forever)	10	0	127
21.	Link T2 timeout (10ms increments)	100	0	6000
22.	Link T3 timeout (10ms increments)	18000	0	65535
23.	AX.25 digipeating (1=enabled, 0=disabled)	1	0	1
24.	Validate callsigns (1=enabled, 0=disabled)	1	0	1
25.	Station ID beacons (2=on, 1=active, 0=off)	2	0	2
26.	CQ broadcasts (1=enabled, 0=disabled)	1	0	1

To display the current settings of the node parameters, use PARMS with no arguments:

**PARMS**

```
LAS:K7WS-1} 50 1 192 255 6 5 3600 64 60 3 3
180 4 4 900 64 10 4 7 10 100 18000 1 1 2 1
```

To change node parameters, use PARMS followed by a series of decimal values in the same sequence described above:

**PARMS \* \* 224 \* 8 6**

```
LAS:K7WS-1} 50 1 224 255 8 6 3600 64 60 3 3
180 4 4 900 64 10 4 7 10 100 18000 1 1 2 1
```

To change a particular parameter, you must enter values for all preceding parameters as well. Entering "\*" in place of a value causes the corresponding parameter value to be left unchanged. If fewer than the full 26 values are entered, the trailing parameters are also left unchanged.

Before using the PARMS command to change node parameters remotely, you must validate your credentials as a control operator by using the SYSOP command...otherwise, the node parameters are left unchanged.

---

## RESET Command

The RESET command allows an authorized control operator to perform a warm-start reset of the NET/ROM firmware. The command is:

```
RESET 32767
```

Any other form of the RESET command is ignored. Before using this command remotely, you must validate your credentials as a control operator by using the SYSOP command...otherwise, the RESET is ignored.

**WARNING:** This command is dangerous, and should be used only as a last resort. The RESET command instantly terminates all links and circuits at the node (including the control operator's own connection).

---

## ROUTES Command

The ROUTES command is used to display or modify the neighbor list of the node's routing table.

To display the node's neighbor list, use ROUTES without any parameters:

```
ROUTES
LAS:K7WS-1} Routes:
> 1 K7WS-11 255 5
> 0 WA7GTU-1 192 17
    0 WA7GTU-2 0 15 !
    0 K7WS-2 via K7WS-4 144 2
    0 WB7BNI-1 192 6
> 0 AA6TN-1 192 27
```

For each neighboring node, the following items are displayed in sequence:

- ">" if active crosslink exists to this neighbor
- port number (0=HDLC port, 1=RS232-port)
- path to this neighbor (callsign + any digits)
- path quality (255 is best, 0 is worst)
- use count (number of routes via this neighbor)
- "!" if this neighbor list entry is locked

To display this information for just one particular neighbor list entry, use ROUTES followed by the port number and path:

```
ROUTES 0 AA6TN-1
LAS:K7WS-1} Routes:
> 0 AA6TN-1 192 27
```

Neighbor list entries may be created automatically as the result of receiving an automatic routing broadcast, or manually by means of the NODES+ command. When a neighbor list entry is first created, it starts out unlocked and with a path quality equal to the default channel quality (see PARMS command). However, the control operator has the ability to "fine-tune" NET/ROM's automatic routing by modifying the path quality values for specific neighbors and by locking these modified entries.

The ROUTES command supports manual modifications to neighbor list entries, but this capability is available only to a control operator who has previously validated his credentials during this connection by successfully executing the SYSOP command.

To modify neighbor list entries, the commands are:

```
ROUTES port nodecall [digicall...] + pathquality
ROUTES port nodecall [digicall...] - pathquality
```

The "+" version locks the neighbor list entry specified by the *port*, *nodecall*, and *digicall* parameters, and sets the path quality of that entry to the value *pathquality* (255 is best, 0 is worst). If there is no entry in the neighbor list that matches *port*, *nodecall*, and *digicall*, a new entry is created, locked, and initialized with the specified *pathquality* and a use count of zero.

The "-" version unlocks the specified neighbor list entry. If its use count is zero, the entry is deleted immediately. Otherwise, the entry remains in the neighbor list and its path quality is set to the value *pathquality*. If the use count of an unlocked neighbor list entry ever becomes zero, the entry is deleted.

The path quality for a neighbor is used by NET/ROM in its calculations of route qualities for all routes through that neighbor. By modifying the path quality using the ROUTES+ command, the control operator can encourage or discourage a node from using paths through a particular neighbor. By setting a neighbor's path quality to zero, the control operator can cause the node to completely ignore the existence of that neighbor, including disregarding the neighbor's routing broadcasts.

*NOTE: The ROUTES command was introduced in NET/ROM version 1.2. On a node using an earlier version, you will get the message "Invalid command".*

---

## SYSOP Command

The SYSOP command allows an authorized control operator to establish his credentials prior to making privileged changes using the IDENT, NODES, PARMS, or ROUTES commands, or executing a RESET command. It uses a randomized validation algorithm, in conjunction with a "password string" previously entered via the node's host terminal. The command is simply "SYSOP", in response to which the node will respond with a list of five random numbers:

```
SYSOP
LAS:K7WS-1} 26 13 54 5 38
```

The control operator must respond by entering the five characters in the correspondingly numbered character positions of the "password string". The random numbers returned by SYSOP will always correspond to valid non-space character positions of the password string. The five required characters may be entered with or without intervening spaces, and must be followed by a carriage-return. There is no acknowledgement of success or failure.

For example, assume the password string is "The quick brown fox jumped over the lazy dog's back 0123456789 times". A valid control operator validation sequence might be:

```
SYSOP
LAS:K7WS-1} 26 13 54 5 38
dolga
```

where the 26th character of the password string is "d", the 13th character is "o", etc. If the validation succeeds, subsequent IDENT, NODES, PARMS and ROUTES update requests and the RESET command are honored; otherwise, they are quietly ignored. When accessing the node from the host terminal or host computer, control operator privileges are automatically granted...there is no need to use the SYSOP command.

---

## USERS Command

The USERS command displays a summary of who is using the node:

```
USERS
BGBEAR:AA6TN-1} NET/ROM 1.3 (708)
Uplink (WA8DED) <--> Host (BGBEAR:AA6TN-1)
Uplink (WB6WEY) <--> Downlink (WB6WEY-15 WB6YMH)
Circuit (LAS:K7WS-1 W6IXU) <--> Downlink (W6IXU-15 WB6UUT)
Circuit (SLC:K7EA-1 W7LN) <--> Circuit (VACA:WA6RDH-1 N6IA)
Circuit (LAS:K7WS-1 K7WS) <~~> CQ (K7WS-15)
Uplink (NK6K)
Uplink (K6IYK) <~~> CQ (K6IYK-15)
Uplink (NK6K-2) <~~> Circuit (SBA:W6AMT-2 NK6K-2)
Circuit (SBA:W6AMT-2 WA6JPR)
```

The heading of the USERS display indicates the version of the NET/ROM firmware in use at the node, and the amount of free RAM space (shown in parentheses, and expressed as a number of 32-byte buffer segments).

After the heading, the USERS display shows the active circuits and links, using the following formats:

- Uplink(fromcall)
- Downlink(fromcall tocall)
- Circuit(node usercall)
- CQ(usercall)
- Host(node)

The "<-->" symbols represent active "patchcords" in the node that connect uplinks, downlinks, circuits, and possibly the host terminal (if any). The "wiggly patchcord" symbol "<~~>" indicates a CQ or connection in-progress. The lines which do not contain the patchcord symbol represent users who are in command mode.

# Host Interface

---

The standard version of NET/ROM allows a terminal to be connected to the TNC. A special version called "Host NET/ROM" supports a computer-oriented interface designed for use with bulletin board systems and similar computer-based network servers. In both versions, the host interface allows the local terminal or computer to issue all the same node commands that a remote user can issue, and also to perform a few privileged operations that are not available remotely.

---

## Terminal Interface

An ordinary ASCII terminal may be connected to the RS232 port of the node's TNC. The host terminal may be used to enter lines of text containing commands or data. Such lines may be up to 256 characters long, including the terminating carriage-return. The terminal's BS or DEL keys may be used to delete the last one or more characters typed on a line; CTRL-U or CTRL-X may be used to delete the entire line.

Lines that start with an ESC character are treated as special host interface commands. The ESC is echoed as "\*", and the next character specifies which function the node is to perform. Valid host commands are B, C, D, I, P, T and Y. (B is in the PK-87/PK-90 version only, and I is in the commercial version only). Each of these commands is described below.

Lines that don't start with an ESC are interpreted as data. If the host interface is connected (see the ESC-C command below), such data lines are passed to the node; otherwise, they are discarded.

### ESC-B — Baud-rates *(PK-87/PK-90 version only)*

The ESC-B command changes the baud rate settings of the TNC. The command accepts a parameter consisting of one or two digits—the first digit sets the HDLC (radio) baud rate, while the second digit (if present) sets the RS232 (terminal) baud rate. Valid digit values are:

- 0 = no change
- 1 = 300 baud
- 2 = 1200 baud
- 3 = 9600 baud

New baud rate settings become effective on the next warm-start (power-up). The default settings correspond to a parameter of '23', which means HDLC=1200 and RS232=9600.

## ESC-C — Connect

---

The ESC-C command connects the host interface to the node. When connected, the host terminal acts like a remote user that has been uplinked to the node. All ordinary node commands (CONNECT, CQ, IDENT, NODES, PARMS, RESET, ROUTES, SYSOP, and USERS) may then be entered as data lines. However, the host connection automatically has control operator privileges, so the SYSOP command is superfluous.

*NOTE: The host terminal connection (like any uplink) is automatically disconnected after 15 minutes of no activity.*

## ESC-D — Disconnect

---

The ESC-D command disconnects the host interface from the node. Data lines from the host terminal are discarded.

*NOTE: For unattended operation, be sure to disconnect the host interface!*

## ESC-I — Identity *(commercial version only)*

---

The ESC-I command sets the node's callsign. It accepts a callsign composed of up to six letters and digits, optionally followed by a hyphen and a numeric SSID in the range 0-15.

ESC-I with no parameter displays the current node callsign.

*NOTE: This command is implemented in the commercial version of NET/ROM only. In the amateur version, the node callsign is "hard-coded" into each ROM. The commercial and amateur versions of NET/ROM are not interoperable with one another.*

## ESC-P — Password

---

The ESC-P command sets the "password string" used by the SYSOP command to validate the credentials of control operators. The phrase may be up to 80 characters long (any excess is ignored), and may include spaces and control characters (except for CR and LF). Upper- and lower-case letters are treated as distinct from one another in the password string.

Note that the random numbers returned by the SYSOP command will always correspond to valid non-space character positions of the password string. For maximum security, it is a good idea to use a password string that is close to the maximum length of 80 characters, and doesn't contain too many spaces.

To set the password to the null string (thereby disabling remote control capabilities altogether), enter ESC-P-LF-CR.

ESC-P with no parameter (i.e., ESC-P-CR) displays the current password string.

---

## ESC-T — TXDELAY

---

The ESC-T command sets the transmitter key-up time delay (TXDELAY), which is the time interval allowed by the node between asserting push-to-talk and starting to transmit data on the HDLC port. ESC-T expects a decimal integer parameter in the range 0-255 which defines the delay in 10-millisecond increments. The default value is 30 (i.e., 300 milliseconds).

ESC-T with no parameter displays the current setting.

---

## ESC-Y— Incoming connections

---

ESC-Y-1 enables the host interface for incoming connections. ESC-Y-0 disables the host interface for incoming connections (default). ESC-Y displays the current host interface state (0 or 1).

*NOTE: For unattended operation, be sure to disable incoming connections!*

---

# Computer Interface

Host NET/ROM is a special version of the firmware which allows the node to support a bulletin board system or other server. Host NET/ROM uses the RS232 port of the TNC to implement a transparent, error-checked, record-oriented interface to the host computer. Host NET/ROM does not support an RS232 TNC-to-TNC interconnect— nodes running Host NET/ROM are necessarily single-channel nodes.

---

## Protocol delimiters

---

The host interface makes use of three reserved characters as protocol delimiters:

HSTX = '82' hexadecimal

HETX = '83' hexadecimal

HDLE = '90' hexadecimal

Note that these are simply the ASCII control characters STX, ETX, and DEL with the high-order bit set.

---

## Blocks

---

All data passed between the host and the node are encapsulated into "blocks" of the form

HSTX	...data...	HETX	checksum
------	------------	------	----------

The data portion of the block may contain any number of bytes of data. Any HSTX, HETX, or HDLE characters embedded in the data are prefixed by HDLE. The checksum is computed by taking the 8-bit binary sum of the data bytes and discarding any carries. The checksum calculation does not include the initial HSTX, the final HETX, or any HDLE prefix characters.

All transactions between host and node consist of a "command block" from host to node followed by a "response block" from node to host. The node never speaks unless first spoken to.

## Command blocks

---

The data portion of a command block consists of:

- an 8-bit binary channel number
- an ASCII command letter
- optionally, variable-length parameters or data

The channel number specifies a logical channel in the range from 0 to 19 (Host NET/ROM presently supports 20 host channels). In addition, channel number 'FF' hex specifies a pseudo-channel used to support monitor mode. The command letter, parameters, and data are described below.

## Response blocks

---

The data portion of a response block consists of:

- an 8-bit binary status byte
- optionally, variable length data

Most commands return a status byte of '00' or 'FF' hex, corresponding to "success" and "failure", respectively. The 'G' and 'S' commands can return other values as well.

### 'B' Command — Baud-rates (PK-87/PK-90 version only)

---

Command: 

ch#	'B'	baudbyte
-----	-----	----------

Response: 

'00' = success
'FF' = failure

'B' sets the baud rate settings of the TNC. *Baudbyte* is an 8-bit parameter composed of two 4-bit fields—the least-significant field sets the HDLC (radio) baud rate, while the most-significant field sets the RS232 (terminal) baud rate. Valid field values are 1, 2, 3 and 0, corresponding to 300-, 1200-, 9600-baud and "no change", respectively. The *ch#* byte is ignored, although it must be valid. New baud rate settings become effective on the next warm-start (power-up). The default settings correspond to a *baudbyte* of '32' hex (HDLC=1200, RS232=9600).

### 'C' Command — Connect

---

Command: 

ch#	'C'
-----	-----

Response: 

'00' = success
'FF' = failure

'C' connects the specified host channel (0-19) to the node—this is logically equivalent to "uplinking" to the node. It normally returns "success" and enqueues a "connected to" link status

message for the channel. If the channel is already connected, it still returns "success" but does not enqueue a link status message. It returns "failure" if the channel number is not valid, or if the specified channel is already occupied by an inbound (user-initiated) connection.

## 'D' Command — Disconnect

---

Command: 

ch#	'D'
-----	-----

Response: 

'00' = success
'FF' = failure

'D' disconnects the specified host channel (0-19) from the node—this is logically equivalent to disconnecting an uplink. It normally returns "success" and enqueues a "disconnected from" link status message for the channel. It returns "failure" if the channel number is not valid, or if the specified channel is not connected in the first place.

## 'G' Command — Get a message

---

Command: 

ch#	'G'	seq#
-----	-----	------

Response: 

0-3 = success	seq#	...data...
'FF' = failure		

'G' dequeues and returns the next message from the specified host channel (0-19, or 'FF' hex for the monitor channel). The status byte indicates what sort of message (if any) is being returned:

- '00' = user data (0-328 bytes)
- '01' = link status "connected to" (callsign)
- '02' = link status "disconnected from" (callsign)
- '03' = link status "connect request from" (callsign)
- 'FF' = nothing enqueued on channel

The *seq#* in the response is echoed back from the *seq#* in the command. The *seq#* is simply a sequential 8-bit binary sequence number assigned by the host for messages dequeued from a particular channel. NET/ROM actually ignores all but the least-significant (odd/even) bit and treats each value as 0 or 1. Messages passed to the host in response to a 'G' command are not actually dequeued and deleted by the node until a subsequent 'G' command is received for the same channel but with a different sequence number. By sending successive 'G' commands with the same channel and sequence numbers, the host can re-read a message any number of times. This permits graceful recovery from transmission errors between the host and the node.

If the 'G' command returns a '00' status byte, the data portion of the response block contains user data (on channels 0-19) or a monitor frame (on channel 'FF'). User data may be 0-256 bytes long. A monitor frame may be 0-328 bytes long.

If the 'G' command returns a '01', '02', or '03' status, the data portion of the response block contains an ASCII callsign (possibly suffixed with a hyphen and SSID).

### 'I' Command — Identity *(commercial version only)*

---

Command: 

ch#	'P'	...callsign...
-----	-----	----------------

Response: 

'00' = success
'FF' = failure

'I' sets the node's callsign. The *callsign* is composed of up to six letters and digits, optionally followed by a hyphen and a numeric SSID in the range 0-15. The *ch#* byte is ignored, although it must be valid.

*NOTE: This command is implemented in the commercial version of Host NET/ROM only. In the amateur version, the node callsign is "hard-coded" into each ROM. The commercial and amateur versions of NET/ROM are not interoperable with one another.*

### 'M' Command — Monitor

---

Command: 

ch#	'M'	'00'=off
		'01'=on

Response: 

'00' = success
'FF' = failure

'M' turns monitor mode on and off. The *ch#* byte is ignored, although it must be valid. When monitor mode is turned on, the node enqueues monitor frames to pseudo-channel 'FF' hex, where they must be retrieved with the 'G' command. Monitor frames consist of raw AX.25 frames with only the HDLC flags and frame-check-sequence stripped off. To provide an intelligible monitor display, the host software must decode the AX.25 header and any higher-level headers. The maximum size of a monitor frame is 328 bytes (256 data plus 72 bytes AX.25 header). Monitoring in no way affects the operation of the regular host channels (0-19).

### 'P' Command — Password

---

Command: 

ch#	'P'	...password...
-----	-----	----------------

Response: 

'00' = success
'FF' = failure

'P' sets the node's password phrase used by the SYSOP command to validate remote control operator access. The *ch#* byte is ignored, although it must be valid. The *password* may be 0-80 bytes long. A null *password* disables all remote control operator capabilities.

## 'Q' Command — Resynchronize

Command: 

ch#	'Q'
-----	-----

Response: 

'00' = success
'FF' = failure

'Q' is used to resynchronize a host interface channel during host startup, shutdown, or after a failure. It disconnects the channel if it is connected, discards all messages enqueued for the channel, and reinitializes the channel's odd/even sequence counters (both incoming and outgoing) to zero.

## 'S' Command — Send a message

Command: 

ch#	'S'	seq#	...data...
-----	-----	------	------------

Response: 

'00' = accepted	seq#
'01' = choke	
'02' = disconnect	
'FF' = failure	

'S' enqueues a message to be sent to the specified channel (0-19). The *seq#* returned in the response is an echo of the *seq#* sent in the command, and is used by the node as an odd/even oscillator in the same fashion described for the 'G' command. If the host sends successive 'S' commands with identical channel and sequence numbers, the node will disregard all but the first one.

'S' returns a '00' status byte if the message is successfully enqueued for the specified channel, and increments its internal sequence counter. Any nonzero status byte indicates that the message could not be accepted.

'S' returns a '01' status byte if the message was not accepted because of a choke condition—either the node has too many messages backlogged for this channel, or the node has insufficient memory space left to buffer this message. The host should generally issue 'G' commands to dequeue any incoming messages, and then retry the 'S' command a bit later (using the same *seq#* value as before).

'S' returns a '02' status byte if the message was not accepted because the specified host channel is not connected. The host should generally discard the outgoing message, and issue 'G' commands to dequeue any incoming messages, one of which is likely to be a "disconnected from" link status message.

'S' returns a 'FF' status byte if *ch#* is invalid (it must be 0-19) or *seq#* is missing.

## 'T' Command — TXDELAY

---

Command: 

ch#	'T'	txdelay
-----	-----	---------

Response: 

'00' = success
'FF' = failure

'T' sets the node's transmitter key-up delay according to the 8-bit binary value *txdelay* which specifies the delay in 10 millisecond increments. The default value is 30 (i.e., 300 milliseconds). The *ch#* byte is ignored, although it must be valid.

## 'Y' Command — Incoming connections

---

Command: 

ch#	'Y'	maxconnect
-----	-----	------------

Response: 

'00' = success
'FF' = failure

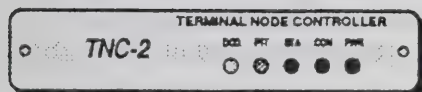
'Y' sets maximum number of incoming (user-initiated) connections to the host that will be allowed by the node, according to the 8-bit binary value *maxconnect* which must be in the range 0-20. The *ch#* byte is ignored, although it must be valid.

Setting *maxconnect* to zero prevents all subsequent incoming connections to the host, and furthermore suppresses enqueueing of "connect request from" link status messages when incoming connections are attempted. As long as *maxconnect* is zero and monitor mode is off, the node can be used safely as a stand-alone NET/ROM node with no host attached. The *maxconnect* parameter reverts to zero and monitor mode reverts to off whenever the node is warm-started (powered-up).

# Installation

This chapter provides a step-by-step checklist for installing a new NET, ROM node. For a hassle-free installation, be sure to follow the instructions carefully.

## TNC-2 Hardware Setup



The following instructions apply to the TAPR TNC-2 terminal node controller or any of the commercially-available TNC-2 "clones" including:

- AEA PK-80
- GLB TNC2A
- MFJ 1270 and 1274
- Pac-Comm TNC-200
- California Digital TNC-2
- Fuji Digital Systems DA-12F and DA-12M
- AIWA APX-25
- TASC0 TNC-20

### Verify PCB revision

NET/ROM requires hardware using TAPR TNC-2 revision 2 (or later) circuitry. Late-model TAPR kits and all commercially manufactured "clones" use this circuitry. However, a few early-model TAPR kits were shipped with the version 1 circuit board, and these must be upgraded with the version 2 modifications available from TAPR.

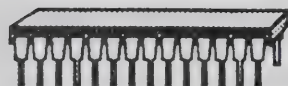
### Increase RAM to 32K

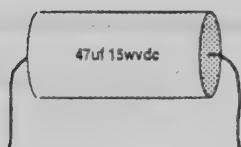
For most installations, NET/ROM should be installed in a TNC with 32K of RAM. It will run with 16K, but will run short of memory under moderate-to-heavy traffic loads.

Most TNC-2s manufactured prior to 1987 were equipped with only 16K of RAM. Inasmuch as the latest version of TAPR firmware requires 32K, however, many TNC-2s have already been upgraded to 32K.

To upgrade a 16K TNC-2 to 32K:

1. Remove U24 and U25 (both are 8K RAMs, type 6264).
2. On the bottom of the circuit board, cut the trace between the center and top positions of JMP12.
3. Add a wire connecting the center and bottom positions of JMP12.
4. Install one 32K RAM IC (type 43256 or 62256) in U25.





## Modify push-to-talk failsafe timer

All tested TNC-2s have a push-to-talk failsafe timer which limits key-down time to approximately 10–12 seconds maximum. This value is too short for NET/ROM use at 1200 baud, and can cause truncation of crosslink transmissions and routing broadcasts. We recommend increasing the failsafe timeout interval to approximately 60 seconds. To do this, simply replace capacitor C31 with a 47 microfarad radial-lead electrolytic capacitor rated at 15 WVDC or higher. Be sure to observe proper polarity.



## Set CPU clock speed to 4.9 MHz

TNC-2's are normally set up for slow CPU clock speed (2.4576 MHz). Although NET/ROM will run reliably at this speed, performance is noticeably improved by changing to high CPU clock speed (4.9152 MHz). Although this is somewhat faster than the rated speed of some TNC-2 parts, we have tested NET/ROM with numerous TNC-2s (including clones from several manufacturers), and none failed to operate reliably at the higher clock speed.

To change a TNC-2 to high clock speed:

1. On the bottom of the circuit board, cut the trace connecting the center and right positions of JMP2.
2. Add a wire connecting the center and left positions of JMP2.

*NOTE: For the MFJ 1270B, the jumpering is different. This may also be true for other TNC-2 "not-quite-clones" so check your TNC documentation carefully.*

## Wire DCD-B to RS232 pin 23

The following modification is required for TNCs that will be used in dual- or multi-channel NET/ROM configurations. Since it does not impair normal operation in any way, we recommend it for all TNCs to be used with NET/ROM...you may want to upgrade to dual-channel operation later.



To make this modification, connect one end of a wire to pin 23 of the RS232 connector. Connect the other end of the wire to pins 1-2-3 of JMP9 (these three pins are already hooked together on the circuit board).

This modification allows the NET/ROM firmware to be configured for multi-channel operation by jumpering RS232 pins 10 and 23 together in the TNC-to-TNC cable.



## Replace U3 op-amp

Most TNC-2s use an LM324 op-amp (U3) to generate bipolar RS232 output signals. However, LM324s do not have sufficient slew rate for reliable operation at 9600 baud. This can be remedied easily by replacing U3 with a faster op-amp. The modification is especially important for dual- or multi-channel nodes where 9600-baud operation is essential.

Simply replace the LM324 at U3 with either a TL074 or TL084 IC (manufactured by Texas Instruments and perhaps others).

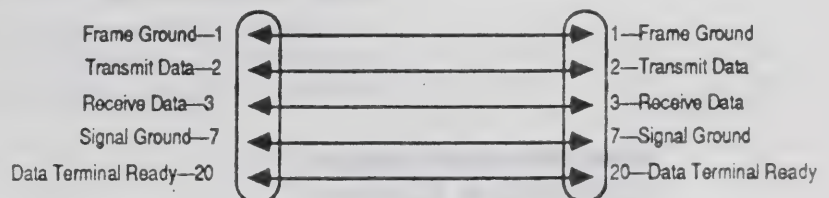
## Set baud-rate switches

Follow the switch-setting instructions in the TNC-2 manual. We have performed extensive reliability testing of NET/ROM at HDLC and RS232 baud rates up to 9600; it runs reliably at these high baud rates even at slow CPU clock speed, although we strongly advise using 4.9 MHz CPU clock for reasons of efficiency.

For dual- or multi-channel operation, we strongly suggest setting the RS232 speed at 9600 baud—it makes a big difference in cross-channel performance! Be sure to upgrade the U3 op-amp, however.

## Connect host terminal

To connect a host terminal to the TNC-2, almost any standard RS232 cable will do (as long as pins 9, 10, and 23 are not used). If you are wiring a special cable for this purpose, wire it as follows:



It is also OK to leave DTR (pin 20) unconnected. Be certain to leave pins 9, 10, and 23 unconnected (at the TNC end, at least).

## Make sure your TNC still works...

Connect a terminal to the TNC, power it up, and make sure that you get a sign-on message and that the unit still appears healthy. If it doesn't, then (1) you have made an error; (2) you have installed a bad IC; or (3) your TNC won't run at the fast CPU clock speed (quite unlikely).



## Finally, install NET/ROM

NET/ROM is distributed in the form of a 27C256 EPROM which simply plugs into the ROM socket (U23) of the TNC-2 in place of the standard TAPR firmware ROM. Be very careful when inserting the new EPROM, making sure that pin 1 is oriented correctly, and that none of the pins are bent under the IC.

For the amateur radio version NET/ROM, the node's amateur callsign is "hard-coded" into each NET/ROM EPROM, and cannot be changed. If you must change the node's callsign, you will have to order a new EPROM.

*NOTE: Be certain to save the original ROM—you may need it if you ever want to recalibrate the TNC modem or to restore regular TNC functionality.*

---

## PK-87/PK-90 Hardware Setup



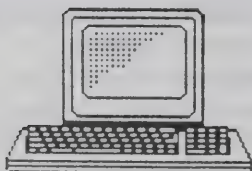
Modify the PK-87 or PK-90 in accordance with the applicable AEA service bulletin. As indicated in the service bulletin, AEA offers both a parts kit and a factory modification service.

For dual-channel installations, on each end of the TNC-to-TNC interconnect cable, add a jumper from pin 4 to pin 6 (in addition to the jumper from pin 10 to pin 23). Multi-channel installations of three or more interconnected PK-87s or PK-90s are not recommended, since the diode coupler illustrated in the NET/ROM manual will not work with the PK-87/PK-90.

The PK-87/PK-90 baud rates are set by software (rather than by the DIP switches used by the TNC-2). The default baud rates used by NET/ROM are 1200 baud for the HDLC (radio) port, and 9600 baud for the RS232 (terminal) port. NET/ROM's host interface includes a command for changing the baud rates.

---

## Software Setup

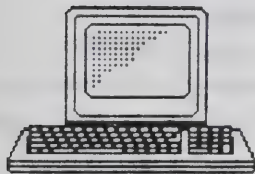


After setting up the TNC hardware, connect a terminal to the RS232 port. Power up the TNC and make sure you see the NET/ROM sign-on message. Then perform the following steps:

### Set or verify the node's callsign

For the amateur radio version of NET/ROM, the hard-coded callsign of the node is displayed in the NET/ROM sign-on message. Make sure it is correct!

For the commercial version of NET/ROM, set the callsign using the ESC-I command described in the "Host Interface" chapter. The node will not transmit unless the callsign has been set.



---

## Connect to the node

Enter ESC-C followed by a carriage-return. This connects the host terminal to the node's command interpreter. The node should respond with a "CONNECTED to..." message.

---

## Set the node's identifier

Use the IDENT command to enter the mnemonic identifier of the node, which can be up to six characters long. Don't use punctuation or non-printing control characters in the identifier, and don't use an identifier that "looks like" a valid amateur call sign. (Suggestion: three-letter airport identifiers make nice node mnemonics.)

---

## Set the password string

Enter ESC-P followed by a "password string" up to 80 characters long. It is best to pick a string that occupies the full 80 characters, or close to it, and doesn't contain too many spaces. The string may contain any ASCII characters except CR and LF...even non-printing control characters are legal. You must remember the password string in order to perform privileged control operator functions remotely. Don't forget it unless you enjoy trips to the site!

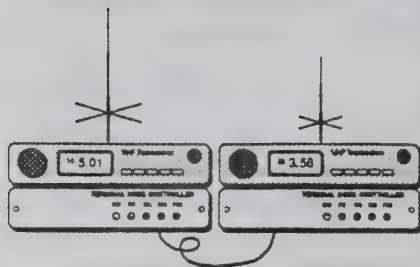
---

## Prepare for unattended operation

Enter ESC-D to make sure the host interface is disconnected. Enter ESC-Y-0 to disable host connections. You should always remember to do these two things before you disconnect the host terminal.

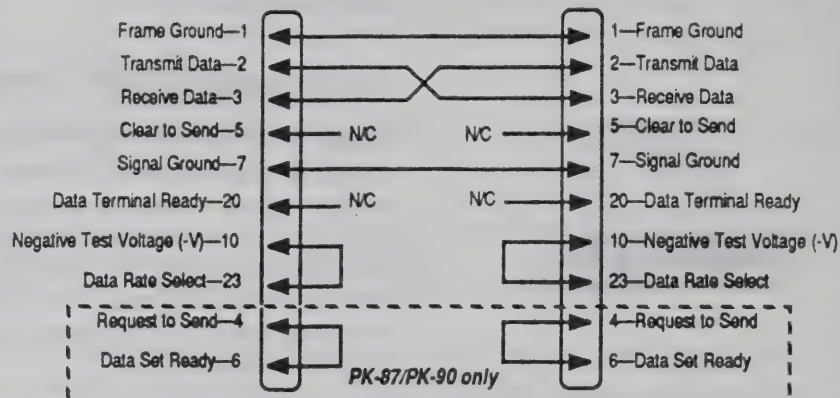
*NOTE: The software setup described above assumes that you are using the standard configuration of NET/ROM that supports a terminal interface. For Host NET/ROM, the setup procedure is analogous, but a computer and suitable host software is required to access the node's host interface.*

## Dual- or Multi-Channel Installations

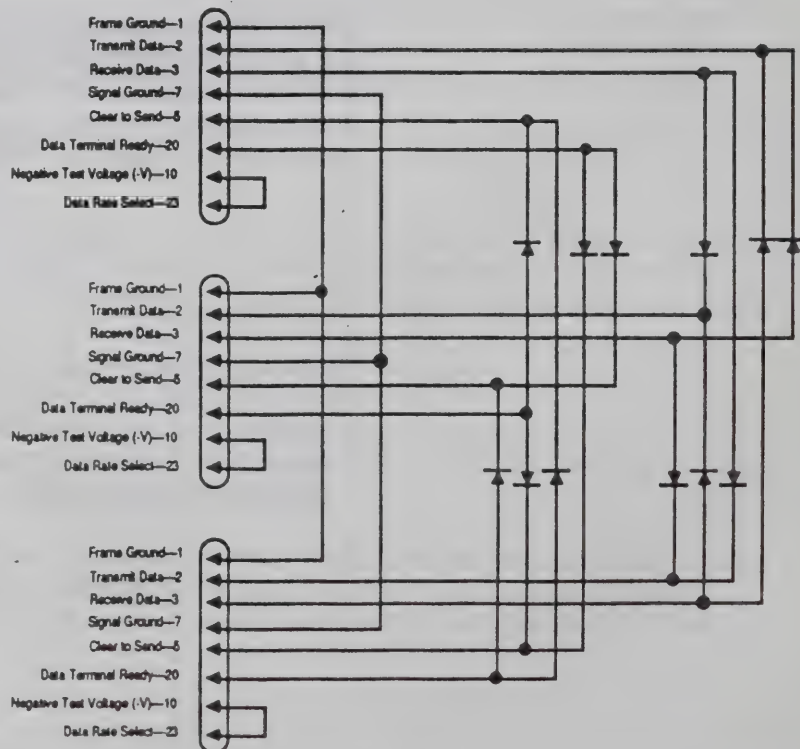


To install a dual- or multi-channel NET/ROM node, you should follow the previously-described steps for each TNC. Make certain that each TNC has a different callsign (often, only the SSID suffix changes). Verify that each TNC is functioning correctly as a single-channel node before attempting to interconnect the TNCs.

For a dual-channel node, simply connect the two TNCs together, using a special RS232 cable wired as shown in the diagram. (Don't try to use the terminal cable—it won't work!)



To interconnect three or more TNCs as a multi-channel node, you will have to make up a diode-matrix coupler. A schematic for a three-channel coupler is shown below. It uses 12 diodes (1N4148 or equivalent). A four-port coupler is similar, but requires 24 diodes, and is probably the maximum practical configuration.



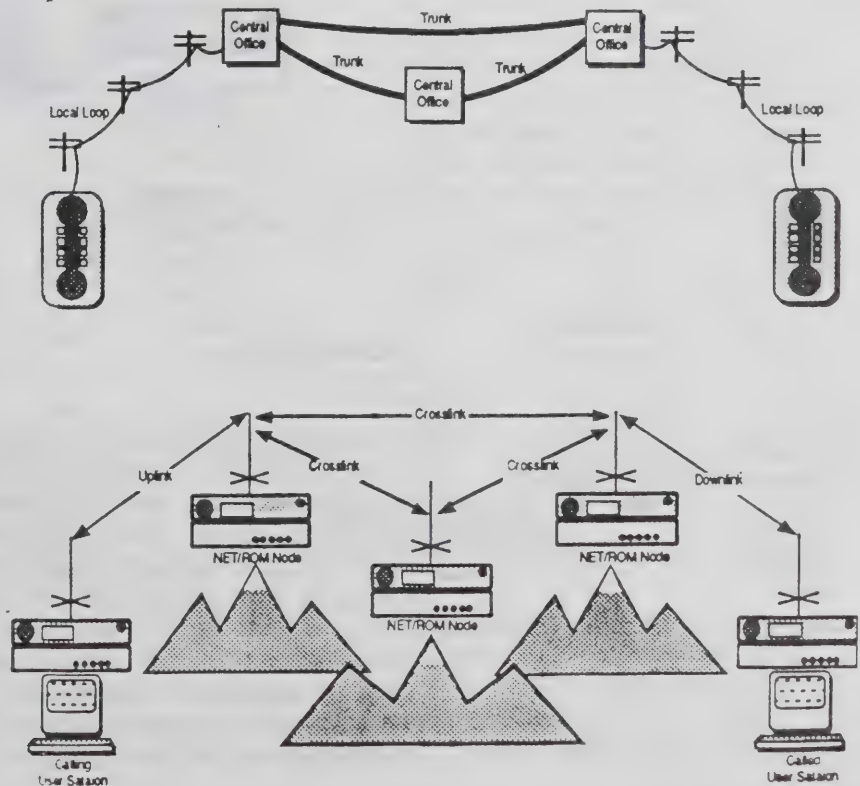
# Theory of Operation

**A**lthough NET/ROM is exceptionally easy to install and use, its inner workings are rather complicated. In this chapter, we examine some of the most important aspects of NET/ROM's internal operation.

## Making Connections

The purpose of any packet radio networking system is to facilitate connections between user stations. Connections in the NET/ROM network are composed of "uplinks", "downlinks", "crosslinks" and "circuits".

To acquire a good understanding of the NET/ROM network, it is helpful to keep in mind that most familiar of networks: the telephone system. In this analogy, user stations correspond to the telephones, and NET/ROM nodes correspond to the central offices. Uplinks and downlinks correspond to the "local loops" that connect each telephone with its local central office, while crosslinks correspond to the "trunks" that connect one central office to another.



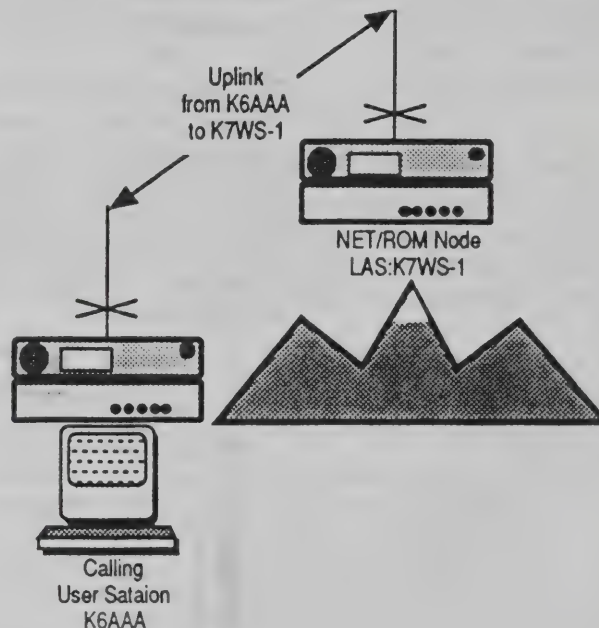
## Links

A "link" is simply an AX.25 connection between a pair of packet radio stations. In this discussion, we are concerned specifically with links that involve a NET/ROM node at one or both ends. Node-to-node links ("crosslinks") always use AX.25v2 protocol. Node-to-user links ("uplinks" and "downlinks") use AX.25v1 if the user's TNC supports it, otherwise AX.25v1.

## Uplinks

An "uplink" is a link between a user station and a node, initiated by the user station. Uplinking to a local node is always the first step a user must take in order to access the NET/ROM network, just as lifting the receiver and waiting for the dialtone is always the first step in accessing the telephone network.

Uplinking is accomplished using the "connect" command of the user's TNC. The node may be addressed either by its callsign or by its mnemonic identifier. An uplink is usually a direct point-to-point connection, but may be digipeated if necessary.

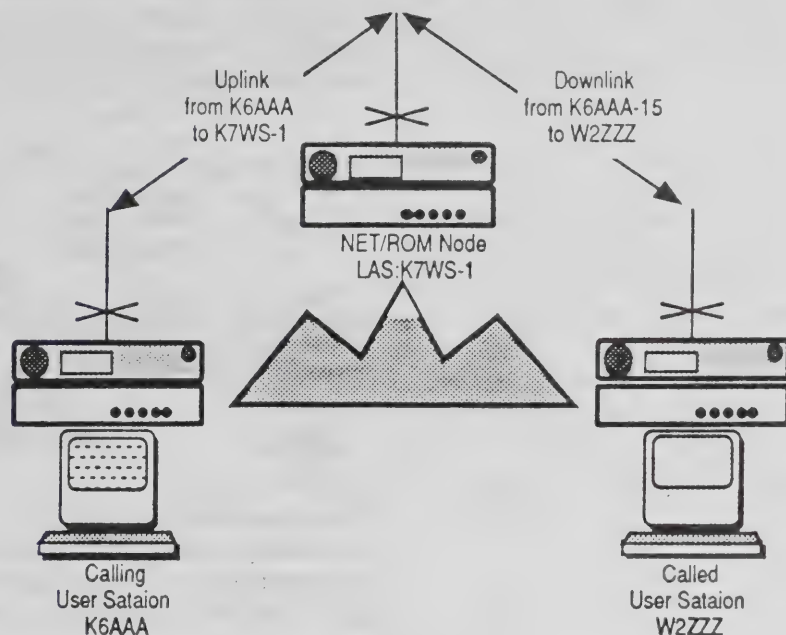


## Downlinks

A "downlink" is a link between a node and a user station (the "called" station), initiated by the node at the behest of another user (the "calling" station) in response to a NET/ROM "CONNECT" command. This is analogous to the telephone central office causing the called subscriber's telephone to ring at the behest of the calling subscriber in response to dialing.

A downlink is usually a direct point-to-point connection, but may be digipeated if necessary. When the node initiates a downlink to the called station, it "adopts" the callsign of the calling station (rather than using its own callsign). This is

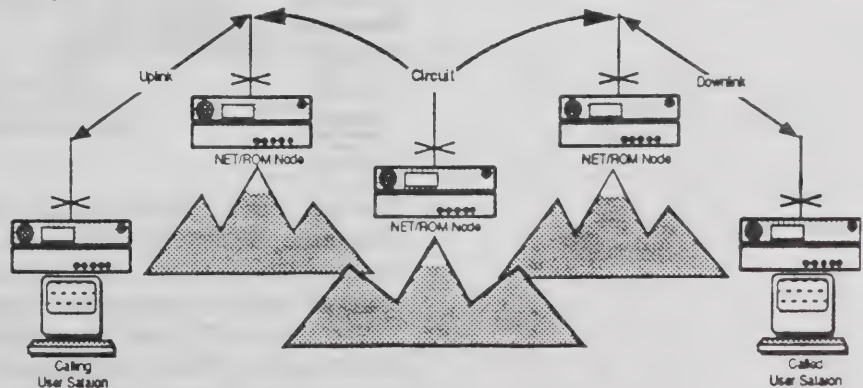
necessary so that the called station can properly identify the calling station, and is especially important when the called station is a bulletin board or other server.



When the node "adopts" the callsign of the calling user, it modifies the SSID (the numeric callsign suffix used by AX.25) from N to 15-N. For example, K6AAA is changed to K6AAA-15; W3ABC-2 is changed to W3ABC-13; and so forth. This is done to prevent the serious protocol problems that would otherwise result if the two user stations could hear one another directly.

## Circuits

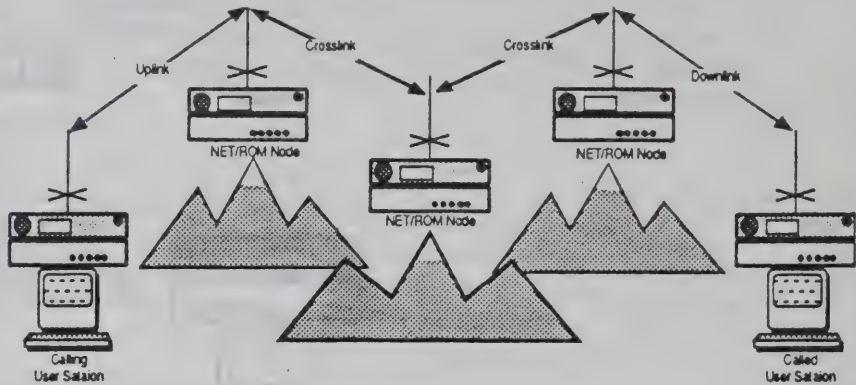
A "circuit" is a transport-layer connection between two nodes. A circuit is established by one of the nodes at the behest of a user in response to a "CONNECT" command. The two nodes are not necessarily adjacent, and may be quite distant. The circuit is automatically routed through intermediate nodes as necessary. Circuits are carried from node to node over crosslinks.



This process is analogous to the establishment of a circuit from one telephone central office to a distant central office when a subscriber dials a long-distance access code and area code. Such a circuit is automatically routed through intermediate telephone switching offices, and is carried from office to office over trunks.

## Crosslinks

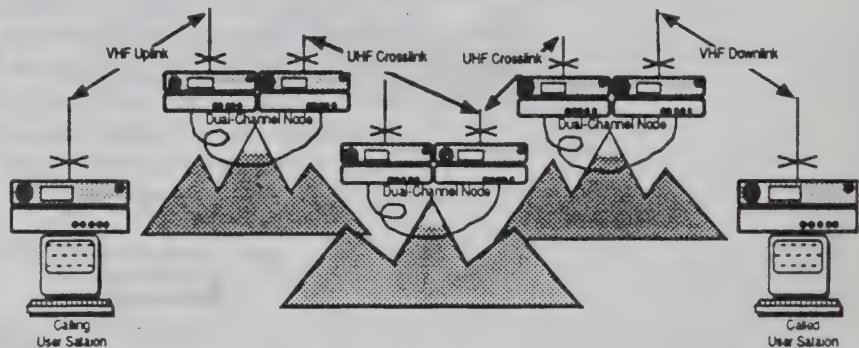
A "crosslink" is a link between two adjacent nodes. A crosslink is usually a direct point-to-point connection, but may be digi-peated if necessary. Crosslinks are set up by the nodes on an as-needed basis during the establishment of circuits.



One crosslink can carry any number of circuits, so it is never necessary to have more than one crosslink between any pair of adjacent nodes. A crosslink remains connected until it has gone idle for 15 minutes, at which point it is disconnected.

## Backbones

To realize NET/ROM's full potential, it is an excellent idea to design a network that minimizes interference between local (uplink/downlink) and long-haul (crosslink) traffic. One good way to accomplish this is to reserve one radio channel exclusively for inter-node crosslink traffic, to provide user access to the nodes on one or more separate channels, and to keep user stations off the inter-node "backbone" channel.



This approach requires nodes that can access two or more channels. NET/ROM supports such multi-channel operation without the need for special multi-port hardware. A dual-channel node, for example, consists simply of a pair of ordinary TNCs (with NET/ROM in each) interconnected with a simple RS232 cable. Each TNC handles traffic on one channel, and cross-channel traffic is passed over the interconnect cable at high speed.

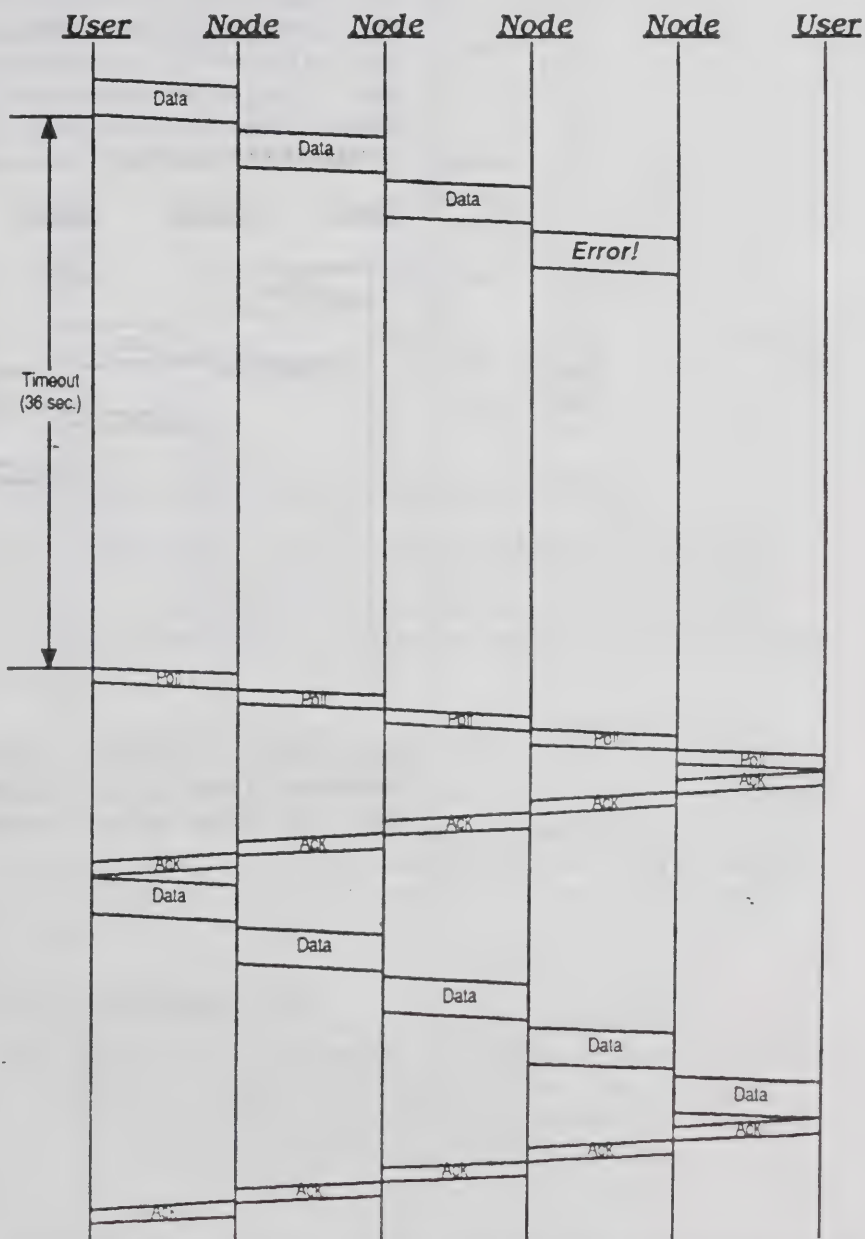
Configuring a three- or four-channel NET/ROM node is very similar, except that a diode coupler is required for proper isolation.

## Digipeating vs. Store-and-Forward

The AX.25 protocol was originally designed for point-to-point (not digipeated) connections. AX.25 was subsequently extended to accomodate one digipeater, and later extended again to allow up to eight digipeaters.

### Multi-hop digipeating

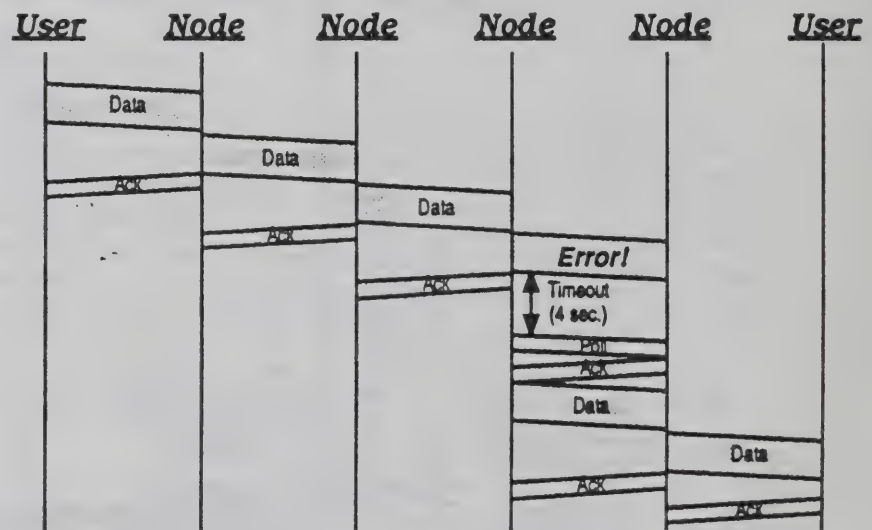
As all experienced packet radio users know, however, AX.25 is practically unusable for communications on paths exceeding two or three "hops". The reason for this is that AX.25 digipeaters do not participate in error control. For an AX.25 packet to traverse a multi-hop path, it must not fall victim to a collision or other error during any of the hops; otherwise, it must be retransmitted by the originating station and start its journey all over again.



The probability that an AX.25 packet can complete its journey successfully deteriorates rapidly as the number of hops increases. For example, it takes five hops to get from San Diego to San Francisco. If the average reliability per hop is 90% (perhaps optimistic), then the probability that a packet will traverse the five-hop path unscathed and be acknowledged (which requires ten error-free hops) is less than 35%. In other words, it will take nearly three tries (on average) to get the packet through successfully. Since the usual timeout for a five-hop path is 36 seconds, the average elapsed time to get the packet through will average about 77 seconds!

## Store-and-forward packet switching

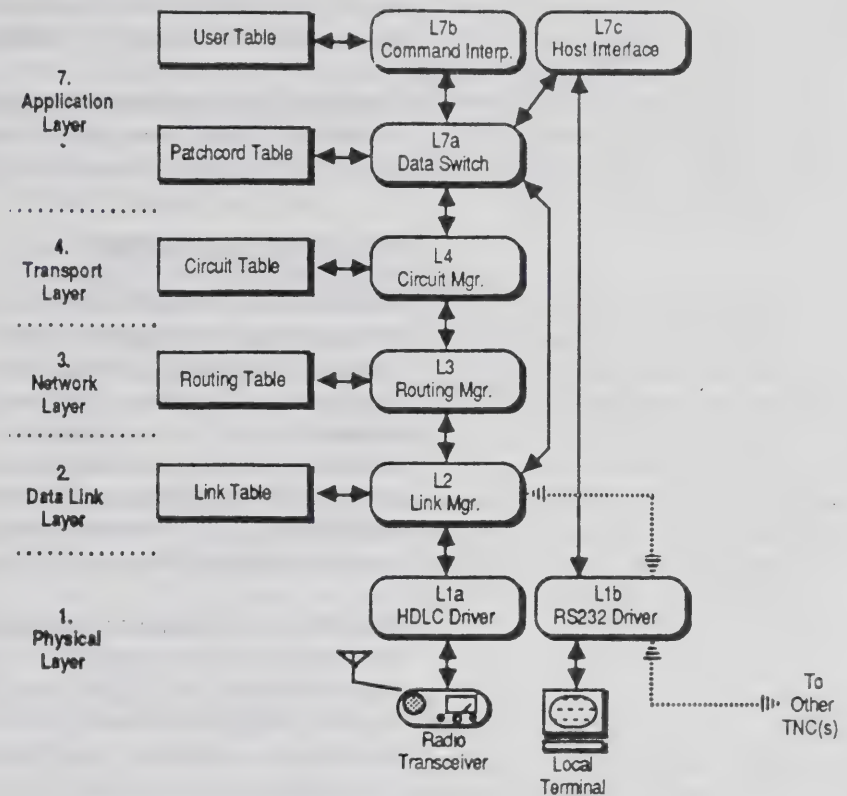
Using NET/ROM nodes instead of ordinary AX.25 digipeaters changes this situation dramatically for the better. When the San Diego user transmits a packet destined for San Francisco, it is received by the local NET/ROM node serving San Diego. That node immediately passes the packet to its neighboring node to the north, and sends an acknowledgement back to the user. This process is repeated five times in all. Whenever a packet is lost due to collision or other error, recovery is handled by just the two adjacent nodes involved.



As a result, the average elapsed time to get a packet through decreases to less than 10 seconds, about an 800% improvement. For longer paths, the payoff is even more dramatic.

# Structure of the Firmware

As shown in the accompanying diagram, the NET/ROM firmware is composed of eight major modules organized closely along the lines of the seven-layer ISO reference model.



## HDLC and RS232 drivers (L1a, L1b)

These are low-level hardware-dependent driver routines that support the two serial ports of the TNC. They include interrupt-service routines written in assembly language. (Almost the entire remainder of NET/ROM is written in the C programming language.)

The RS232 driver determines whether the RS232 port is attached to a terminal or to one or more other TNCs by looking at DCD-B (wired to RS232 pin 23, as described in the installation instructions). If a terminal is attached, the RS232 driver interfaces with the Host Interface module (solid arrows); if another TNC is attached, the RS232 driver interfaces with the Link Manager (dotted arrows).

## Link Manager (L2)

The Link Manager manages all AX.25 links (uplinks, downlinks, and crosslinks), and maintains a link table to keep track of currently-active links. This module includes most of the capabilities of a normal AX.25 TNC, including digipeating.

The AX.25 protocol provides for a "Protocol Identification" (PID) field in the header of information frames. NET/ROM identifies information that it sends over node-to-node crosslinks with a special PID value ('CF' for the amateur radio version), while information sent to or from users over uplinks and downlinks use the standard PID value ('FO'). The Link Manager passes incoming inter-node 'CF'-frames to the Routing Manager, whereas it passes incoming user 'FO'-frames directly to the Data Switch.

The Link Manager monitors traffic on each active link. If no information has been passed over a particular link for 15 minutes, that link is automatically disconnected.

### **Routing manager (L3)**

---

The Routing Manager maintains the routing table, and handles the automatic routing of crosslink traffic. It examines the network header of arriving frames to determine the call sign of the destination node. If a frame is destined for this node, it is passed to the Circuit Manager. Otherwise, the routing table is searched for the destination node, the next node along the route to that destination is determined, a crosslink to that node is established if none already exists, and then the frame is passed back to the Link Manager for crosslinking.

The Routing Manager also facilitates manual and automatic updates to the routing table. It supports the manual update capabilities of the NODES and ROUTES commands, which can be used locally or remotely by an authorized control operator. It also issues automatic hourly routing broadcasts, and analyzes broadcasts received from other nodes to see if automatic routing table updates are needed.

### **Circuit manager (L4)**

---

The Circuit Manager handles the initiation, disconnection, and end-to-end error and flow control for transport-layer circuits between nodes, and maintains a circuit table to keep track of currently-active circuits.

The Circuit Manager receives incoming frames from the Routing Manager. It analyzes the transport header of each frame, using a "sliding window protocol" to detect missing, duplicate, and out-of-sequence frames. It enforces circuit-layer flow control to protect the network against disproportionate loading by any one circuit. After validation, the incoming frames are passed to the Data Switch. For outgoing frames, the flow is just the reverse.

New circuits are initiated only when explicitly requested by a user (via the CONNECT command). The Circuit Manager monitors traffic on each active circuit. If no information has been passed over a particular circuit for 15 minutes, that circuit is automatically disconnected.

## Data switch (L7a)

The Data Switch module acts as a switchboard within the node. It maintains a table of "patchcords" which establish two-way linkages between links, circuits, the Command Interpreter, and the Host Interface.

When a user first connects to a node (via an uplink, or via a circuit from another node), the Data Switch initially connects his uplink (or circuit) to the Command Interpreter. If the user then issues a CONNECT command to request a downlink, circuit, or host connection, then the Data Switch "patches him through" as appropriate.

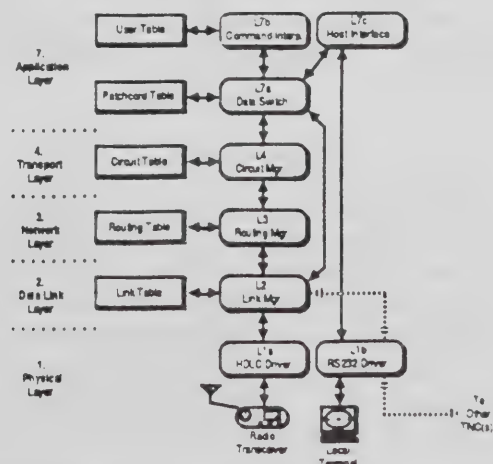
## Command interpreter (L7b)

The Command Interpreter analyzes and executes user commands, and diagnoses any resulting errors that may occur. When a user first enters a command, the Command Interpreter establishes a user task and makes a corresponding entry into its user table. The user remains in command mode (and his task remains active) until he disconnects, executes a successful CONNECT command, or executes a CQ command and receives a reply. The Command Interpreter supports nine commands: CONNECT, CQ, IDENT, NODES, PARMS, RESET, ROUTES, SYSOP, and USERS.

## Host interface (L7c)

The Host Interface supports a local terminal attached to the node. It permits the local terminal to access all of the same capabilities that a remote user can access. In addition, it supports some specialized "host commands" available only from the local terminal. For example, it allows local entry of the password string used to validate control operator credentials.

There are two alternative versions of the host interface module. The standard configuration of NET/ROM provides a host interface designed to talk to a terminal. An alternative configuration called Host NET/ROM provides a record-oriented host interface designed to talk to a bulletin board or other computer-driven server.

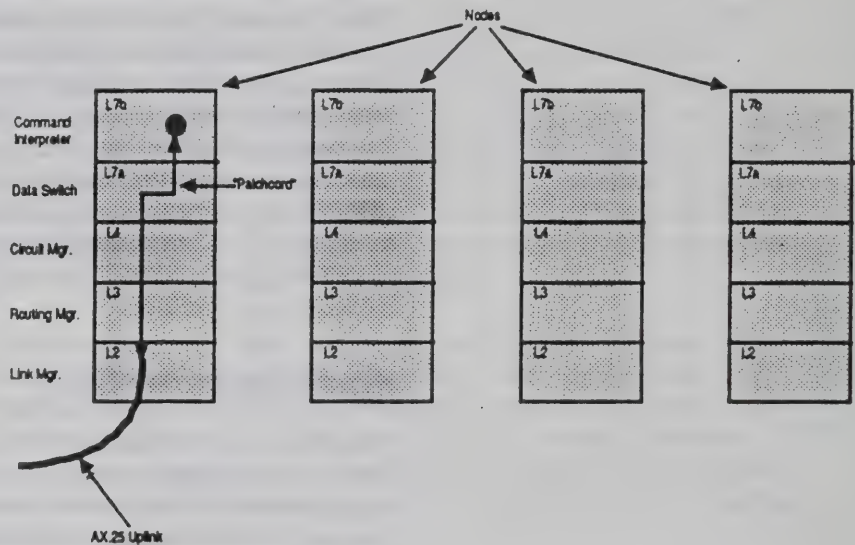


## Internal Information Flow

Now, let's take a look at the step-by-step process that takes place inside the nodes when a user accesses the network...

### Uplink phase

First, the user uplinks to his local node. The Link Manager acknowledges the user's connect request, and makes a new entry in its link table. Since it is an uplink, incoming information frames from the user are passed directly to the Data Switch. Initially, the Data Switch always patches an uplink through to the Command Interpreter. The Command Interpreter makes a new entry in its user table, and starts interpreting the user's commands.

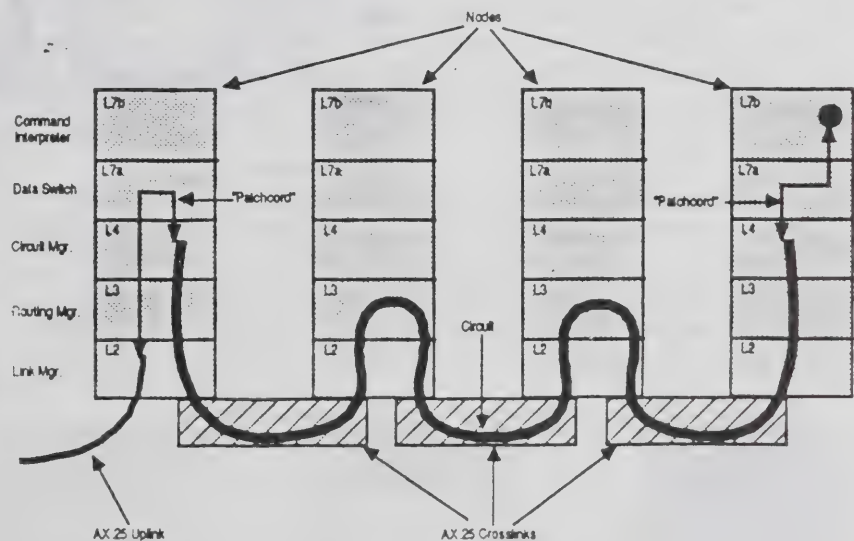


## Crosslink phase

Next, the user issues a CONNECT command that specifies the callsign or identifier of a distant node. The command is passed from the Link Manager to the Data Switch to the Command Interpreter. The Command Interpreter asks the Circuit Manager to establish a new circuit to the distant node. If this succeeds, then the Command Interpreter asks the Data Switch to set up a "patchcord" between the user's uplink and the newly-created circuit. (If the circuit cannot be established for some reason, then the Command Interpreter give the user a diagnostic message.)

To set up the new circuit, the Circuit Manager creates a transport-layer connect request message, and sends it to the Routing Manager. The Routing Manager uses its routing table to determine the next node en-route to the ultimate destination. There probably is a crosslink to that node already; if not, the Routing Manager asks the Link Manager to establish one. Then, the Routing Manager passes the connect request message to the Link Manager for crosslinking.

The frame is received by the Link Manager of the next node. Since it came from a crosslink, it is passed to the Routing Manager. The Routing Manager sees that the ultimate destination hasn't been reached yet, consults its routing table to find the next en-route node, sets up a crosslink if one doesn't already exist, and passes the message on. This continues until the message reaches its destination.

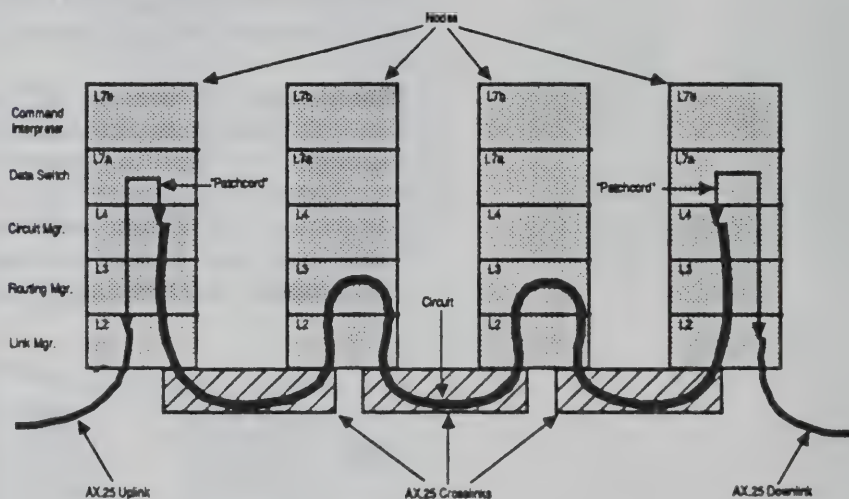


At the destination, the message passes from the Link Manager to the Routing Manager to the Circuit Manager. The Circuit Manager creates a transport-layer connect acknowledge message, and passes it back to the originating node via the same snake-like path (in reverse). Thereafter, any information received over the new circuit by the Circuit Manager at the distant node is passed to the Data Switch. Initially, the Data Switch patches it through to the Command Interpreter.

## Downlink phase

Now, the user is talking to the distant node, and he issues a CONNECT command that specifies the callsign of another user. The command passes from the uplink, through the patchcord in the local node to the circuit, through the circuit (via that snake-like path through the intermediate nodes and crosslinks) to the distant node, and through another patchcord to the distant node's Command Interpreter.

This time, the distant Command Interpreter analyzes the CONNECT command and asks the Link Manager to establish a new downlink to the specified user station. If this succeeds, then the Command Interpreter asks the Data Switch to shift its patchcord so that it connects the circuit to the new downlink. From that point on, the calling user has a transparent connection to the called user. (Once again, if the downlink cannot be established for some reason, the distant Command Interpreter will issue an appropriate diagnostic message.)



# Automatic Routing Mechanism

When a user asks one node for a circuit to a distant node, his CONNECT command specifies the callsign or mnemonic identifier of the destination node, but the routing is handled automatically by NET/ROM. Automatic routing is handled by the Routing Manager, and is controlled by its routing table.

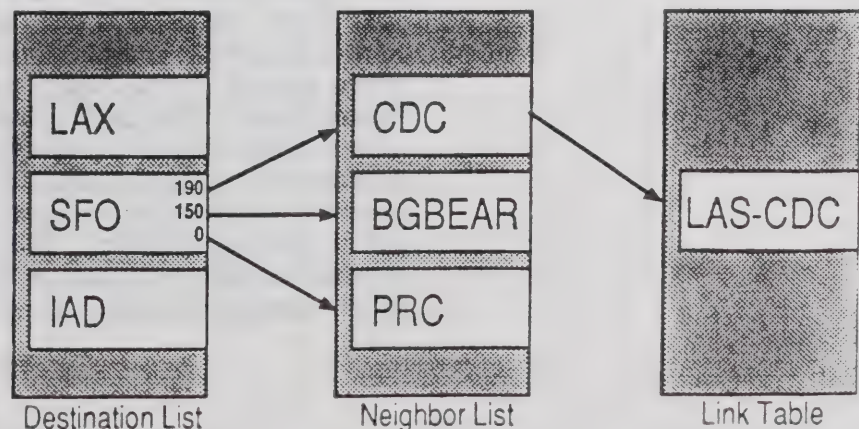
## Routing table

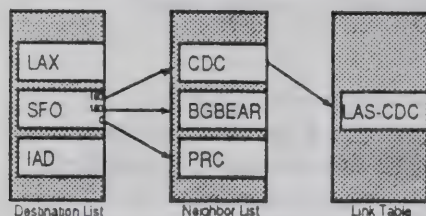
The routing table within a node contains a list of all other nodes "known" to the node, together with their mnemonic identifiers. A user can ask to see this list by issuing a parameterless NODES command. The routing table can keep track of hundreds of other nodes (limited only by the size of available memory and any constraints imposed by the control operator).

If a user wants to connect to an *especially* distant node, it is possible that his local node doesn't know of it (i.e., it is not listed in the local NODES display). In this case, he can use CONNECT to obtain a circuit to a known node nearer the desired destination, and then issue another NODES command to get a list of the nodes known to *that* node. This process can be repeated if necessary.

For each known node, the routing table can contain up to three alternate ways to route traffic to that node. The node knows the quality of each alternate route, and always attempts to use the "best" (i.e., highest quality) route to a destination. However, if a node or path becomes unusable due to equipment failure or poor propagation, the Routing Manager automatically switches to an alternate route (if available) to circumvent the outage. Such routing changes are handled dynamically, usually without disrupting circuits in progress.

The routing table maintained by each node consists of two dynamically allocated threaded lists: the *destination list* and the *neighbor list*. The destination list contains an entry for every other node "known" to this node. (This is the list displayed by the NODES command.) The neighbor list contains an entry for only those "neighboring" nodes to which this node can crosslink *directly*.





For each node in the destination list, the routing table keeps track of up to three *routes* to that destination node. In this context, a route simply identifies a neighboring node that is a step closer to the ultimate destination. For each route, the destination list maintains a *quality* value in the range 255 (best) to 0 (worst). Routes are maintained in sorted order by quality, and NET/ROM always attempts to use the highest-quality route available. It also keeps an *obsolescence count*, which enables NET/ROM to purge a route from its routing table when it has become unuseable and remained so for a protracted period of time.

Observe that the routing table does *not* contain the entire path to each known destination (this is unnecessary, and would require too much memory), but just a list of neighboring nodes that are reasonable choices for a next step enroute to the destination. You can ask to see the routes to a particular destination node by using a NODES command that specifies the callsign or mnemonic identifier of the destination.

## Routing table updates

To enable the network to incorporate new nodes and to adapt to changing propagation and equipment outages, NET/ROM provides a mechanism for automatic update of routing information on a distributed basis. Once an hour, each node automatically broadcasts a list of all other nodes which it knows how to reach—for each such node, the broadcast includes the callsign, mnemonic identifier, and the optimum route and its quality. When the broadcast is received by neighboring nodes, they automatically make any necessary additions, deletions, and modifications to their routing tables, and incorporate the revised information into their own hourly broadcasts. Thus, whenever a new node is placed on-line, knowledge of the new node automatically propagates throughout the network within a few hours. If a node or route becomes unuseable, that information also propagates automatically.

In addition, NET/ROM permits the control operator to make manual changes to a node's routing table. The NODES+ and NODES- commands allow the control operator to add, delete, and modify any entry in the routing table. The ROUTES+ and ROUTES- commands allow the control operator to define the path quality between a node and each of its immediate neighbors, and permits marginal paths to be locked out altogether. These manual update capabilities are available to the control operator both at the node site and via remote control. To make routing table changes remotely, the control operator simply connects to the node, validates his control operator authority by means of the SYSOP command, and then updates the routing information as necessary. The SYSOP command uses a randomized verification algorithm that makes it extremely difficult for an unauthorized user to masquerade as a control operator.

## Route quality analysis

For each route in its routing table, NET/ROM maintains a route quality value in the range 255 (best) to 0 (worst). This allows it to keep alternate routes in order of quality, and to select the best available route to a destination.

A route quality value can best be visualized as a fraction (the value divided by 256) which quantifies the speed and reliability of a particular route in comparison to a theoretically perfect route (an infinitely fast and perfectly error-free path) of quality 256. For example, a route of quality 230 can be thought of as being "90% perfect" ( $230/256 = 0.90$ ).

The quality of each crosslink path used by a NET/ROM node is established by the control operator of the node. Path quality to specific neighboring nodes may be individually set using the ROUTES+ command, and the default path quality for each channel may be set using the PARMS command. As a starting point, we suggest the following values to be used as path quality parameters:

<u>Channel Description</u>	<u>Quality</u>	<u>% Perfect</u>
9600-baud RS232 wire interconnect (2-port)	255	99%+
9600-baud RS232 satellite interconnect (2-port)	252	98%
9600-baud RS232 wire interconnect (3-port)	248	97%
9600-baud HDLC isolated internode backbone	240	94%
1200-baud HDLC isolated internode backbone	224	88%
1200-baud HDLC user-accessed channel	192	75%
300-baud HDLC HF channel	128	50%
Known loopback or route of unknown quality	0	?

The quality value of zero is reserved by NET/ROM to identify a route of unknown quality or a known loopback route.

The quality of a multi-segment route is simply the product of the qualities of each individual segment, where quality values are treated as fractions with an implicit denominator of 256. For example, a four-segment route that consists of two 9600-baud RS232 interconnect segments (quality 255) and two 1200-baud HDLC backbone segments (quality 224) has a calculated quality of 194:

$$\frac{255}{256} \times \frac{255}{256} \times \frac{224}{256} \times \frac{224}{256} = \frac{194}{256}$$

Quality calculations are rounded to the nearest 256th.

---

## Some Other Facilities

---

### Error and flow control

NET/ROM uses the standard AX.25v2 link-layer packet radio protocol for crosslinks between neighboring nodes, as well as for links from each node to its local users. Normal AX.25v2 error handling is used to assure error-free transmission, and normal AX.25v2 flow control is used to control network congestion.

In addition, NET/ROM incorporates a transport-layer "sliding window protocol" to provide end-to-end error and flow control for each virtual circuit. End-to-end error control is necessary to protect against lost, duplicate, or out-of-sequence frames resulting from node failures and dynamic routing changes. End-to-end flow control is necessary to protect the network against disproportionate loading by any one circuit.

The transport-layer protocol used by NET/ROM is similar in concept to the familiar AX.25, but is somewhat more sophisticated. For example, it can accept out-of-sequence frames and re-sequence them internally. It can also selectively request a repeat of a missing frame without requiring retransmission of all succeeding frames.

---

### Deferred disconnect

A classic problem in packet radio occurs at the end of a connection when one station wants to disconnect, but not before making certain that the other station has successfully received the first station's concluding information frames. NET/ROM solves this dilemma in a straightforward fashion.

If two stations are connected to one another via the network and one of the stations disconnects, NET/ROM automatically maintains its connection to the other station until all in-transit information frames have been successfully delivered to that station. NET/ROM disconnects only after all in-transit information has been delivered, or after 15 minutes has elapsed without any "forward progress" in delivering such information.

---

### Station identification

In order to conform with FCC regulations concerning station identification, each NET/ROM node identifies itself every 10 minutes. The station identification is broadcast as an AX.25 UI-frame addressed to "ID" and containing the text "Network node" plus the node's mnemonic identifier (if it has one) in parentheses. The control operator can make such broadcasts conditional upon node activity, or disable them altogether if desired, by means of a PARMS parameter.

## Digipeating

---

A NET/ROM node supports the functions of an ordinary AX.25 digipeater. This allows digipeater owners can upgrade their sites to NET/ROM nodes without notifying or inconveniencing the user population—the users won't notice any change unless they happen to monitor a station-id broadcast or try to connect to the node (expecting a "busy" message).

Furthermore, each multi-channel node is also a multi-port digipeater. Each channel is assigned a different callsign. Often, the same basic callsign will be used, but with different SSID suffixes for each frequency. (For example, K7WS-1 on VHF and K7WS-11 on UHF.) Cross-frequency digipeating is requested simply by including both callsigns as digipeaters (e.g., "CONNECT W2ZZZ via K7WS-1,K7WS-11,...").

If a node is assigned a mnemonic identifier, that identifier is also recognized as an AX.25 "alias". It can be used in lieu of the callsign for purposes of digipeating ("CONNECT W2ZZZ via LAS,CDC,FRISCO,SLC") or uplinking ("CONNECT LAS").

The control operator can disable the node's digipeating capabilities, if desired, by means of a PARMS parameter.

## Control operator validation

---

Authorized control operators can make manual updates to routing table entries with the NODES and ROUTES commands, modify various node parameters with the IDENT and PARMS commands, and warm-start the node with the RESET command. To do these things remotely, a control operator must first validate his credentials by means of the SYSOP command—otherwise, the RESET command and the update functions of the IDENT, NODES, ROUTES and PARMS commands are locked out.

Because packet communications between the control operator and the node can be easily monitored (and imitated) by any user, an ordinary password scheme cannot provide effective validation. Consequently, the SYSOP command utilizes a randomized validation algorithm that makes it quite difficult for an unauthorized user to masquerade as a control operator.

A password string up to 80 characters long is entered into the node by an on-site operator via a local terminal. The password string cannot be changed remotely. When a remote operator enters the SYSOP command, the node replies with a list of five random numbers (each in the range 1-80). The operator must then enter the five characters that correspond to those numbered character positions in the password string. The five correct characters must be entered before control operator privileges are granted.

A user who monitors this procedure can learn only five characters of the password string. He would have to monitor a large number of such procedures before he could learn enough of the password string to have any reasonable chance of a successful masquerade. To make things harder, the password string may contain non-printing control characters, if desired. To make things even harder, the command interpreter does not report whether or not a SYSOP validation procedure has succeeded or failed—consequently, an especially cautious control operator could perform two or three SYSOPs, giving false answers to all but one of them, and thereby giving misleading information to anyone who might be monitoring. Of course, it also is prudent to change the password string from time to time whenever the control operator visits the site.

# Node Notes

---

**H**ere is a collection of helpful suggestions compiled as a result of many months of watching the actual use of NET/ROM in the field, and taking note of the mistakes that users and control operators seem to make most often.

Users cannot be expected to make effective use of the network without some education in the workings of NET/ROM. At minimum, users should read the "Guided Tour", "Commands", and "Notes for Users" sections of this manual. Control operators should disseminate this basic information to their users in the form of hardcopy or a downloadable text file.

---

## Notes for Users

### Checking if you have a decent path

---

You are uplinked to your local NET/ROM node, and now you want to connect to a distant node. Before you do, it's a good idea to check whether or not you have a decent path to your intended destination.

First, issue a parameterless NODES command to obtain a list of known destination nodes. Check to see whether the node you want is in this list. If it isn't, you can't get there.

If it is, issue another NODES command specifying the callsign or identifier of the node you want. For instance, if you want to get to the Salt Lake City node:

```
NODES SLC
LAS:K7WS-1} Routes to SLC:K7EA-1
      126 6 1 K7WS-11
> 108 5 0 WA7GTU-1
      0 6 0 AA6TN-1
```

You will see up to three routes to the node. If you see the symbol ">", it points to the route you should look at. Otherwise, look at the first route in the list.

The route in question will have a sequence of three numbers—in this case, "108", "5", and "0".

The first number ("108") is the *quality* of the route. The higher it is, the better the route is (theoretically). Routes with qualities less than 100 may not be useable. If the route quality is zero, it's definitely best not to attempt to connect to the node.

The second number ("5") is the *obsolescence count*, and indicates how recent the routing information is. This value is normally set to 6 whenever a routing broadcast is received that

indicates the route is still OK, and decremented each time such a routing broadcast is *not* received as expected. The count is usually 5 or 6—a lower value often means the route has become marginal or has disappeared entirely. If the count is *zero*, it indicates that the route has been manually locked for some reason.

The third number is always either “0” or “1”. If it is “1”, it indicates that the route involves a cross-over to another channel of a multi-channel node—often to a “backbone” channel. Unlike the first two numbers, this one doesn’t give you any information about the quality of the path.

---

## Don't repeat commands

---

If you have sent a command to a distant node and see no response, *do not* reissue the command! Lack of response probably means that the network is heavily congested. It may also mean that the route to your destination has gone bad, and NET/ROM is trying to set up an alternate route. In any case, repeating the command will just make matters worse.

Your command is still somewhere enroute—it is *not* lost. Be patient...eventually you will either receive a response or (if the path has failed completely) be disconnected.

---

## Don't backtrack

---

Have you ever connected to a distant node, tried to reach some station there, gotten a “busy” or “failure”, and then decided to connect to another, closer node? If you did, your data was routed all the way to the distant node and back again...which was probably *not* what you intended!

The correct procedure is to disconnect your TNC, re-uplink to your local node, and then connect to the desired destination node.

---

## Aborting a CONNECT or CQ command

---

If you issue a CONNECT command and then discover that you misspelled the callsign, don't panic. You don't have to disconnect and start over again. Simply reissue the CONNECT command correctly.

An in-process CONNECT command is aborted immediately when you send any other command to the node. The same is true of an unanswered CQ command.

---

## Sending very long frames

To achieve best efficiency using NET/ROM, you may wish to set your TNC's maximum frame length parameter ("PACLEN") to 236 bytes or less. Here's why. The maximum allowable AX.25 information frame is 256 bytes. However, since NET/ROM inter-node protocols impose an additional overhead of 20 bytes, the transport-layer information field is limited to 236 bytes.

User frames longer than 236 bytes are automatically fragmented into two transport frames by the originating node, and reassembled again into one long frame by the destination node. This fragmentation and reassembly is completely transparent to users of the network, so the only disadvantage of sending long frames (>236 bytes) is some additional overhead.

---

## Using multiple uplinks

Since many TNCs now support multiple AX.25 connections, you may wish to establish multiple simultaneous uplinks to your local NET/ROM node. This can be done successfully, but some precautions are in order. The AX.25 protocol is not capable of supporting multiple simultaneous connections in which both originator and destination callsigns are identical. Consequently, to achieve multiple concurrent uplinks from your TNC to a node, you must employ one of the following tactics:

1. If your TNC is capable of supporting independent originator callsigns ("MYCALLs") on each channel, then you can establish different callsigns for each channel (typically, your call with SSIDs -0, -1, -2, etc.) and then uplink with impunity. Since each uplink will have a unique originator callsign, there will be no protocol ambiguity. This is the best approach...but unfortunately, many TNCs are incapable of assigning independent originator callsigns to each channel.
2. If your local NET/ROM node (let's say it's "K7WS-1") has a mnemonic identifier (let's say it's "LAS"), then the node will actually respond to seventeen different names: its callsign (K7WS-1) plus its identifier with any valid SSID (LAS, LAS-1, LAS-2,..., LAS-15). Consequently, you can establish multiple concurrent uplinks to the node as long as you are careful to call the node by a different name when establishing each uplink.

---

# Notes for Control Operators

---

## Fixed limits

The current version of NET/ROM has the following fixed limits:

- Maximum number of links per node: 25
- Maximum number of circuits per node: 20

---

## Using watchdog timers

Some digipeater operators have been known to incorporate a "watchdog timer" circuit to prevent the possibility of a TNC hangup due to software failure. The simplest approach is to rig up a 555 timer that resets the TNC every 10 or 15 minutes.

Trying to use such a "brute force" approach on a NET/ROM node would be disastrous. Resetting a NET/ROM node causes instant termination of all uplinks, downlinks, and crosslinks to or from that node. *Don't even consider it!*

A somewhat more sophisticated approach is to set up a retriggerable timer triggered by the TNC's push-to-talk output. The idea is to reset the TNC only if PTT has not been activated for some period of time. This approach will probably work OK with NET/ROM as long as the timeout period is at least 15 minutes or so. The automatic station-identification broadcasts every 10 minutes should prevent such a timer from firing unless something goes terribly wrong. If you use this approach, *don't turn off the unconditional station-identification feature!*

The TNC hardware and NET/ROM firmware have shown themselves to be extremely reliable, and consequently the use of such watchdog timers is almost never necessary or warranted.

---

## Remember, it's *radio*!

Extensive field experience with hundreds of nodes has made it clear that 95% of all NET/ROM problems are *analog*, not digital: marginal propagation, poor sites, local RF interference, bad antennas, mistuned cavities, overdeviation, slow squelch action, misaligned modems, and so forth.

While NET/ROM is often capable of bypassing paths that are temporarily down, it can't do anything to fix paths that are permanently poor. In fact, because NET/ROM tries so vigorously to make the best of a bad situation, it tends to create even more congestion when there are no reliable paths available.

To improve network operation, it is almost always better to concentrate on the *radio* aspects of packet radio than anything else.

## Locking-out unusable paths

Although NET/ROM's automatic routing system generally does its job well, it occasionally needs help from the control operator in adapting to local conditions. One of the most important control operator responsibilities is to lock-out marginal or one-way paths that may otherwise confuse NET/ROM's routing algorithms.

NET/ROM updates its routing tables automatically based upon periodic routing broadcasts that it receives from neighboring nodes. Implicit in this technique is the assumption that the node can communicate with any node that it can hear. This is usually true, but not always. Once in a while, you may encounter a marginal path that allows an occasional routing broadcast to be heard but is too poor to be used for crosslinking. Alternatively, you may encounter a neighboring node that cannot hear your node despite the fact that your node can hear it fine.

NET/ROM allows the control operator to lock-out such unusable paths, and it is important to do so. Use the ROUTES+ command to create a locked neighbor-list entry with a path quality of zero. This tells NET/ROM not to route traffic to that neighbor, and not to pay attention to routing broadcasts from that neighbor.

## Assigning path quality values

NET/ROM bases its routing decisions on a numeric *quality* score that it assigns to each route in its routing table. NET/ROM's route quality calculations are based on path quality values assigned by the control operator of each node.

The quality of a path is a value from 0 (worst) to 255 (best). Path quality to specific neighboring nodes may be individually assigned using the ROUTES+ command. The default path quality for each channel (HDLC and RS232) may be set using the PARMS command.

Channel quality depends upon baud-rate, congestion, and many other factors. It is up to the control operator to assign reasonable quality values to the various available crosslink paths in order to allow NET/ROM to make the most efficient routing choices. As a starting point, we suggest the following values to be assigned to channel quality parameters:

<u>Channel Description</u>	<u>Quality</u>	<u>% Perfect</u>
9600-baud RS232 wire interconnect (2-port)	255	99%+
9600-baud RS232 satellite interconnect (2-port)	252	98%
9600-baud RS232 wire interconnect (3-port)	248	97%
9600-baud HDLC isolated internode backbone	240	94%
1200-baud HDLC isolated internode backbone	224	88%
1200-baud HDLC user-accessed channel	192	75%
300-baud HDLC HF channel	128	50%
Known loopback or route of unknown quality	0	?

The default path quality values are 192 for channel 0 (HDLC), and 255 for channel 1 (RS232). Be sure to change these values (using the PARMS command) if they are not appropriate to your particular installation. For example, if your node's HDLC port is interfaced to a high-quality backbone channel, you must increase the channel 0 quality. Likewise, if your node's RS232 port is interconnected to other nodes using a satellite link or diode couples, you must decrease the channel 1 quality parameter. *Don't neglect to set these parameters!*

## Setting up a digipeated crosslink

---

Crosslinks between neighboring nodes are normally direct AX.25 connections. However, NET/ROM is also capable of using crosslinks involving one or two intermediate digipeaters. Digipeated crosslinks are rather undesirable—ideally, the digipeaters should be upgraded to nodes—but they are sometimes necessary during the early phases of network development.

A digipeated crosslink must be set up *manually* by the control operators of both nodes. Suppose the control operators of nodes CONN:W1AAA and DEL:W3CCC decide to establish a crosslink using an intermediate digipeater W2BBB. The two operators should agree on a reasonable path quality for this digipeated crosslink: 100 might be a reasonable value. Then, each operator must make a manual routing table entry. The W1AAA control operator issues the command:

```
NODES W3CCC + DEL 100 0 0 W3CCC W2BBB
```

and the W3CCC control operator issues the command:

```
NODES W1AAA + CONN 100 0 0 W1AAA W2BBB
```

These commands will create permanent locked routing entries, which must be deleted with NODES- commands when they are no longer needed.

## Avoiding locked routing entries

---

Locked routes are entered using the NODES+ command and specifying an obsolescence count of zero. The previous example of a digipeated crosslink is a good example of why this capability was included in the design of NET/ROM. However, experience shows that control operators tend to use locked entries more often than they should, and often do more harm than good in the process.

A locked routing entry is especially harmful when there is a failure of a node or path to which the entry refers. Since the routing entry is locked, NET/ROM will keep trying to use the failed node or path indefinitely. Furthermore, such situations often result in the development of routing loops.

*Avoid the use of locked routing entries wherever possible.*

---

## Disabling automatic routing updates

---

The control operator can disable various portions of NET/ROM's automatic routing update mechanism by using the PARMs command. Setting parameter #2 (worst quality for auto-updates) to zero causes NET/ROM to ignore incoming auto-routing broadcasts. Setting parameter #5 (obsolescence count initializer) to zero inhibits the automatic deletion of obsolete routes. Setting parameter #6 (obsolescence count minimum to be broadcast) to 255 inhibits broadcast of auto-routing information for destinations other than the node itself. Setting parameter #7 (auto-update broadcast interval) to zero inhibits outgoing auto-routing broadcasts altogether. So, for example, you can disable all automatic routing functions by zeroing parameters #2 and #7.

We suggest, however, that you think very carefully before disabling any of the automatic routing functions of a node. Under normal circumstances, the automatic routing mechanisms of NET/ROM do an excellent job of keeping routing table information up-to-date. Furthermore, each NET/ROM node has a "responsibility" to the other nodes in the network to process and broadcast automatic routing information. Disabling the automatic routing functions of one node may affect the routing tables of many other nodes. Therefore, you should probably consider disabling NET/ROM's automatic routing function only under extraordinary circumstances—for example, to protect your node and the rest of the network against "pollution" from a malfunctioning node broadcasting corrupt routing information.

---

## Validating amateur callsigns

---

The amateur radio version of NET/ROM includes a callsign validation algorithm whose purpose is to protect the network against invalid amateur callsigns in commands, link headers, routing table entries, etc. To qualify as a valid amateur callsign, a callsign must meet the following criteria:

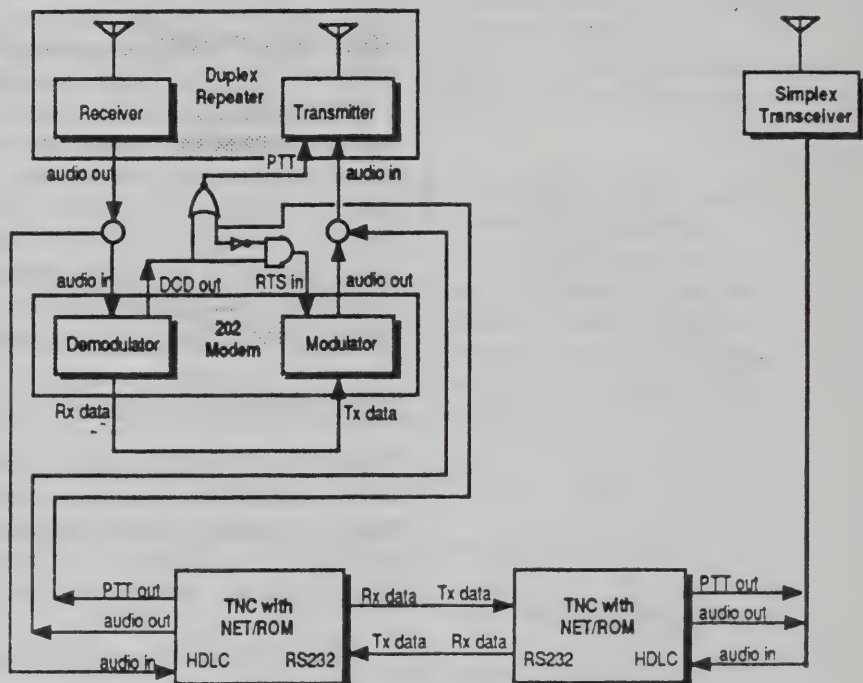
1. Length of callsign (excluding SSID) must be between 4 and 6 characters.
2. All characters must be letters or digits (no punctuation or control characters).
3. The callsign must contain either 1 or 2 digits.
4. The last character of the callsign must be a letter (not a digit).
5. The SSID suffix (if present) must be in the range 0 to 15.

In practice, one of the most useful functions of such callsign validation is to prevent uplinks from improperly-initialized user TNCs which are using a bad callsign (e.g., "NOCALL", "PK64", or blank).

The PARMS command includes a parameter that allows a control operator to defeat this callsign validation if desired. However, we strongly advise against disabling callsign validation in NET/ROM, and we disclaim any responsibility for problems that occur in amateur nodes operating with callsign validation disabled.

## Interfacing with a duplex repeater

In high-density areas with lots of congestion on packet user frequencies, a duplex packet repeater can improve throughput and reduce collisions significantly in comparison to an ordinary simplex digipeater. A dual-port NET/ROM node can be readily interfaced with such a repeater in such a way as to give users on the duplex repeater access to the rest of the normal simplex network. The following diagram illustrates one way in which this can be accomplished.

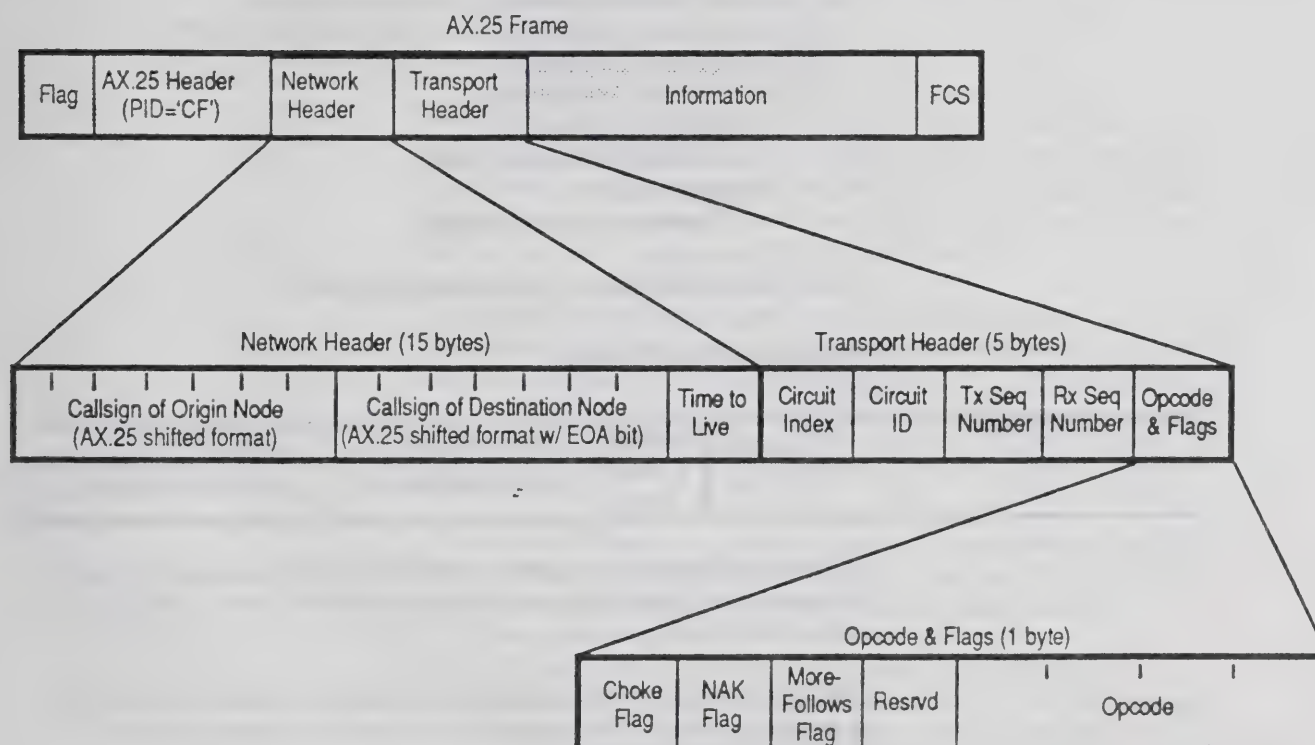


# Protocols

This chapter describes the higher-level protocols employed by NET/ROM. It assumes a knowledge of the underlying AX.25v2 link-layer protocol. [A detailed description of AX.25v2 is available from the American Radio Relay League, Newington, CT 06111, USA.]

## Structure of Inter-Node HDLC Frames

The HDLC frame structure used by NET/ROM for its inter-node crosslinks is illustrated in the following diagram:



Each frame consists of a standard AX.25 link header, followed by a 15-byte network header and a 5-byte transport header. The network header contains the information needed for automatic routing, while the transport header supports end-to-end error and flow control for circuits.

In amateur radio service, NET/ROM identifies its inter-node frames with an AX.25 protocol identifier byte (PID) of 'CF' hex. In commercial service, various other PID values are used.

# Transport Layer (End-to-End) Protocol

The Circuit Manager implements a conventional "sliding window protocol" in order to provide end-to-end flow and error control for each transport circuit. The protocol is similar to AX.25, but with these major differences:

1. The receive window size is negotiated, and is usually greater than 1.
2. Message sequence numbers are 8-bit fields, allowing window sizes up to 127 frames.
3. Selective NAKing is supported.

Six transport-layer message types are supported:

- Connect Request
- Connect Acknowledge
- Disconnect Request
- Disconnect Acknowledge
- Information
- Information Acknowledge

These are described in greater detail below.

## Connect request

My Circuit Index	My Circuit ID	(Unused)	(Unused)	Opcode = 'x1'	Propose Window Size	Callsign of Originating User ( AX.25 shifted format)	Callsign of Originating Node ( AX.25 shifted format)
------------------------	---------------------	----------	----------	------------------	---------------------------	---	---

A Connect Request is used to initiate a new transport circuit between the originating node and the destination node, on behalf of the originating user.

The *circuit index* is the subscript of a circuit table entry in the originating node. The *circuit ID* is a serial number used to qualify the circuit index, in order to eliminate any possible ambiguity in identifying the circuit. The *proposed window size* specifies the maximum receive window size (in frames) that the originating node is prepared to accommodate.

## Connect acknowledge

Your Circuit Index	Your Circuit ID	My Circuit Index	My Circuit ID	Opcode = 'x2'	Accept Window Size
--------------------------	-----------------------	------------------------	---------------------	------------------	--------------------------

A Connect Acknowledge is used to respond to an incoming Connect Request. If the high-order bit of the opcode byte is set, it indicates that the Connect Request is being refused; otherwise, it is being accepted. The *accepted window size* indicates the negotiated size of the receive window for this circuit, and

will never exceed the *proposed window size* of the Connect Request.

## Disconnect request

Your Circuit Index	Your Circuit ID	(Unused)	(Unused)	Opcode = 'x3'
--------------------------	-----------------------	----------	----------	------------------

A Disconnect Request is used to request the termination of a transport circuit, and may be sent by the node at either end of the circuit.

## Disconnect acknowledge

Your Circuit Index	Your Circuit ID	(Unused)	(Unused)	Opcode = 'x4'
--------------------------	-----------------------	----------	----------	------------------

A Disconnect Acknowledge is used to acknowledge a Disconnect Request.

## Information

Your Circuit Index	Your Circuit ID	Tx Sequence Number	Rx Sequence Number	Opcode = 'x5'	Information
--------------------------	-----------------------	--------------------------	--------------------------	------------------	-------------

An Information message is used to pass user information across a transport circuit.

Because AX.25 frames are limited to 256 bytes and the combined network and transport header overhead totals 20 bytes, the maximum size of the information field is 236 bytes. NET/ROM automatically fragments and reassembles any user-supplied link-layer information frames that exceed 236 bytes in order to meet this constraint.

Each Information message also serves as a "piggybacked" Information Acknowledge. The *Tx Sequence Number* identifies the current information, and the *Rx Sequence Number* specifies the next incoming information expected.

If the *choke flag* is set (bit 7 of the opcode byte), it indicates that this node cannot accept any more Information messages until further notice. If the *NAK flag* is set (bit 6 of the opcode byte), it indicates that a selective retransmission of the frame identified by the *Rx Sequence Number* is being requested. If the *more-follows flag* is set (bit 5 of the opcode byte), it indicates that the information is a fragment of a long information frame, and must be reassembled with one or more following information messages by the destination node.

## Information acknowledge

---

Your Circuit Index	Your Circuit ID	(unused)	Rx Sequence Number	Opcode = 'x6'
--------------------------	-----------------------	----------	--------------------------	------------------

An Information Acknowledge is used to acknowledge incoming Information messages. The *Rx Sequence Number* specifies the next incoming information expected.

If the *choke flag* is set (bit 7 of the opcode byte), it indicates that this node cannot accept any more Information messages until further notice. If the *NAK flag* is set (bit 6 of the opcode byte), it indicates that a selective retransmission of the frame identified by the *Rx Sequence Number* is being requested.

---

## RS232 Interconnect Protocol

For multi-channel nodes, information is passed among the interconnected TNCs via their RS232 ports. The interconnect operates using the same link-, network-, and transport-layer protocols as normal HDLC crosslinks, except that a simple asynchronous variant of HDLC is employed.

Each frame is preceded with an ASCII STX (instead of an HDLC flag), and terminated by an ETX plus a one-byte checksum (instead of an HDLC frame-check sequence). Any embedded STX, ETX, or DLE characters within the body of the frame are prefixed by a DLE character (this takes the place of normal HDLC "bit stuffing").

When two TNCs are connected using the recommended TNC-to-TNC cable, communications between the two TNCs is full-duplex and extremely fast (especially at 9600 baud).

When three or more TNCs are interconnected using the recommended diode-matrix coupler circuit, intercommunications between the TNCs is essentially half-duplex and utilizes CSMA/CD arbitration. Consequently, do not expect performance to be quite as spectacular as it is with the dual-channel configuration.

# Routing

This chapter provides a detailed description of the internal workings of NET/ROM's automatic routing mechanism.

## Routing Table Structure

The routing table maintained by each node consists of two dynamically allocated threaded lists: the *destination list* and the *neighbor list*. The destination list contains an entry for every other node "known" to this node—this is the list displayed by the NODES command. The neighbor list contains an entry for only "neighboring" nodes to which this node has a *direct* crosslink—this is the list displayed by the ROUTES command.

*Destination List*

Destination list linkage	Pointer to next destination
	Pointer to previous destination
Is there already an active route to this destination?	
Mnemonic identifier of this destination	
Callsign of this destination	
Which route to use? (1, 2, or 3)	
How many routes are there to choose from? (1-3)	
Route #1	Quality of this route
	Obsolescence count
	Pointer to neighbor list entry
Route #2	Quality of this route
	Obsolescence count
	Pointer to neighbor list entry
Route #3	Quality of this route
	Obsolescence count
	Pointer to neighbor list entry
Information queued for this destination	Pointer to first info frame
	Pointer to last info frame

*Neighbor List*

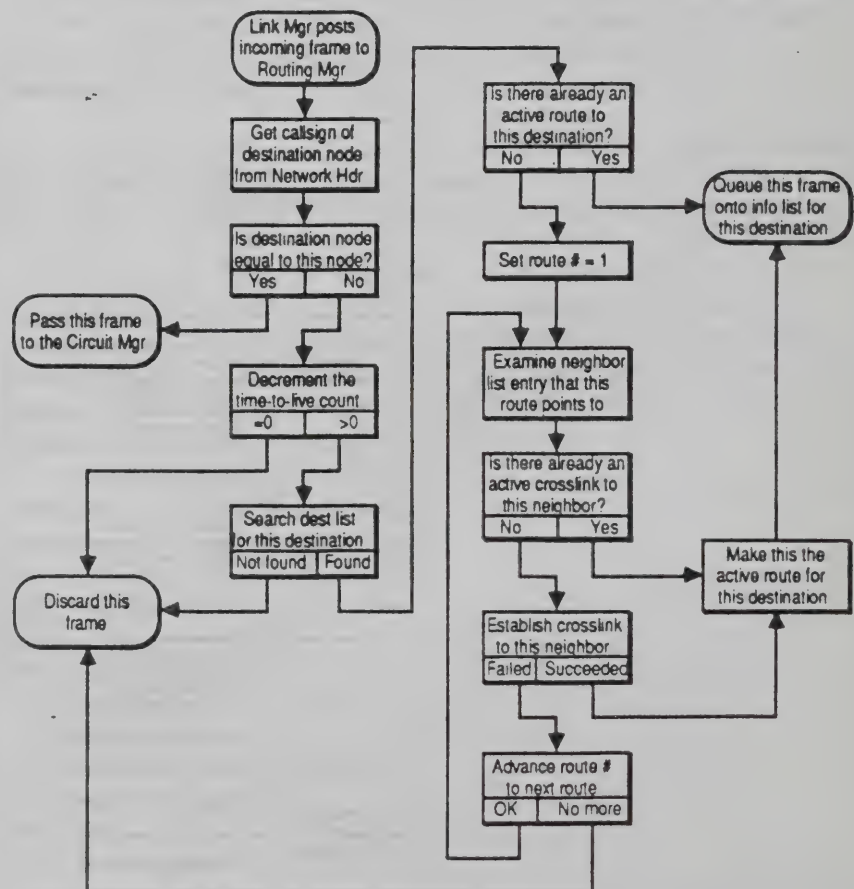
Neighbor list linkage	Pointer to next neighbor
	Pointer to previous neighbor
Callsign of this neighbor	
Digipeaters to reach this neighbor (max 2)	
Which port? (0=HDLC port, 1=RS232 port)	
Path quality to this neighbor	
Is this neighbor-list entry locked?	
How many routes point to this neighbor?	
Pointer to link table (crosslink to this neighbor)	
(Reserved)	

For each node in the destination list, the routing table up to three routes to that destination node. In this context, a route simply identifies a neighboring node that is a step closer to the ultimate destination. For each route, the destination list maintains a *quality* value which quantifies the relative desirability of each route. NET/ROM maintains route quality as an integer which ranges from 255 (best) to 0 (worst). Routes are maintained in sorted order by quality, and NET/ROM always attempts to use the highest-quality route available. It also keeps an *obsolescence count*, which enables NET/ROM to purge paths from its routing table when it has become unuseable and remained so for a protracted period of time.

Observe that the routing table does *not* contain the entire path to each known destination (this is unnecessary, and would require too much memory), but just a list of neighboring nodes that are reasonable choices for a next step enroute to the destination. You can ask to see the various routes to a particular destination node by using a NODES command that specifies the callsign or mnemonic identifier of the destination.

## Routing Algorithm

The Routing Manager analyzes the network header of each incoming crosslink-frame, and determines how to route that frame. The routing algorithm is straightforward, and is summarized in the following (somewhat simplified) flow diagram:



## Automatic Routing Table Updates

Each node makes a periodic broadcast of information from its routing table in order to provide the basis of routing table updates at neighboring nodes. The broadcast is normally made once an hour, although the frequency may be changed with the PARMS command by the control operator.

The routing broadcast takes the form of one or more AX.25 UI-frames tagged with a special PID ('CF' hex in the amateur radio service). The source callsign identifies the broadcasting node, and the destination callsign is "NODES". The information contents of each frame has the following structure:

Signature 'FF' hex	Mnemonic ident of sending node (6 bytes)	Callsign of destination node (7 bytes)	Mnemonic ident of destination node (6 bytes)	Callsign of best- quality neighbor (7 bytes)	Best- quality value	(more destinations)
-----------------------	--	--	--	--	---------------------------	---------------------

Repeat for each known destination  
(up to 11 per UI frame)

Up to eleven destinations are packed into each UI-frame, and the node broadcasts as many such frames as required to send its entire routing table.

When a node receives an auto-routing broadcast UI-frame from one of its neighboring nodes, it analyzes the contents and makes appropriate updates to its own routing table. This process is more complex than one might think. The receiving node utilizes a series of heuristic rules to keep the size of its routing table manageable and to try to avoid "loops" and other undesirable routes. Here is a summary of the most important rules used in processing auto-routing update broadcasts:

1. If the worst quality for auto-updates parameter (set by PARMS) is zero, all auto-update broadcasts are ignored.
2. If the UI-frame does not have the proper PID or signature byte, the frame is ignored.
3. The neighbor list is searched for an entry corresponding to the node that originated the broadcast. If no such entry exists, a new one is created and assigned a default path quality in accordance with the channel quality parameter (set by PARMS) appropriate to the channel (HDLC or RS232) over which the broadcast was received.
4. A direct route is assumed to exist to the node that originated the broadcast. The quality of this route is taken from the path quality in the neighbor-list entry.

5. For each destination node listed in the broadcast, an indirect route is assumed to exist via the node who originated the broadcast. The quality of this route is calculated by combining the route quality from the broadcast with the path quality to the neighbor node as defined in the neighbor-list entry. The qualities are multiplied, normalized, and rounded as shown in the following formula:

$$\text{routequality} = [(\text{broadcastquality} \times \text{pathquality}) + 128] / 256$$

6. An indirect route is considered to be a trivial loop if the callsign of the best-route neighbor node in the broadcast matches the callsign of the receiving node. Trivial loop routes are assigned a quality value of zero, so they are used only as a last resort. Quality-zero routes are never included in outgoing auto-routing broadcasts.
7. Only the three highest-quality routes to a destination are retained.
8. Any route with quality less than the *worst quality for auto-updates* parameter (set by PARMS) is ignored.
9. If the number of entries in the destination list is greater than or equal to the *max destinations* parameter (set by PARMS), then no additional destinations will be added.

Each route in the routing table has an *obsolescence count* which is initialized to the *obsolescence count initializer* parameter (set by PARMS) whenever the route is added or updated as the result of an auto-routing broadcast. The count is also reinitialized to this value whenever the route is actually used successfully. The default initializer value is 6. Periodically, NET/ROM scans through the routing table and decrements the obsolescence count of every route in the table—this scan occurs with the same frequency as routing broadcasts, normally once each hour. If the obsolescence count of a route is decremented to zero, the route is deleted from the routing table.

A routing table entry created by the automatic routing update mechanism can never have an obsolescence count of zero, since such an entry is automatically purged from the table when its count reaches zero. When a route is entered into the routing table manually with the NODES+ command, however, it is possible to set the route's obsolescence count to zero. This has special significance: it marks the route as *locked*. Such a locked route will never be updated or deleted by the auto-update mechanism. It can, however, be deleted manually with the NODES- command.

# Parameters

---

**H**ere is a detailed description of the 26 NET/ROM parameters that the control operator can change using the PARMS command.

## 1. Maximum destination list entries

---

*(default=50, minimum=1, maximum=400)*

Defines the maximum allowable number of destinations in the destination list of the node's routing table. Each destination consumes 32 bytes of RAM. The control operator can use this parameter to limit the amount of RAM that is allocated to the routing table, thus ensuring that sufficient space remains for frame buffering.

## 2. Worst quality for auto-updates

---

*(default=1, minimum=0, maximum=255)*

Defines the poorest route quality that will be automatically added to the node's routing table. The control operator can use this parameter to limit the automatic routing update function to accept only higher-quality routes. In addition, the automatic update function can be disabled altogether by setting this parameter to zero.

## 3. Channel 0 (HDLC) default path quality

---

*(default=192, minimum=0, maximum=255)*

Defines the default quality of the radio channel connected to the node's HDLC port. The control operator should set this parameter to an appropriate quality value in accordance with the speed, reliability, and congestion anticipated on the channel. The default value of 192 is appropriate for a 1200-baud user-accessible frequency...if the actual channel quality is better (e.g., a UHF backbone frequency) or worse (e.g., an HF link), the parameter value should be changed accordingly.

## 4. Channel 1 (RS232) default path quality

---

*(default=255, minimum=0, maximum=255)*

Defines the default quality of the TNC-to-TNC interconnect channel connected to the node's RS232 port. The control operator should set this parameter to an appropriate quality value in accordance with the speed, reliability, and congestion anticipated on the channel. The default value of 255 is appropriate for a 9600-baud two-modem interconnect cable...if the actual channel quality is worse (e.g., a three- or four-port interconnect, or a satellite link), the parameter value should be changed accordingly.

---

## 5. Obsolescence count initializer

---

*(default=6, minimum=0, maximum=255)*

Defines the initial value given to the obsolescence count of a route that has been newly added or updated by the node's automatic routing table update mechanism. The obsolescence count of a route is decremented once each auto-update broadcast interval. However, such periodic decrementing of route obsolescence counts can be disabled altogether by setting this parameter to zero.

---

## 6. Obsolescence count minimum to broadcast

---

*(default=5, minimum=1, maximum=255)*

Defines the minimum obsolescence count threshold below which a route will not be included in the node's automatic routing broadcasts. The purpose of this threshold is to prevent the node from broadcasting "stale" routing information to other nodes. Under normal circumstances, this parameter should be assigned a value no greater than the value of parameter #5 (obsolescence count initializer); if it is greater, the node's broadcasts will include no destinations other than itself.

---

## 7. Auto-update broadcast interval (seconds)

---

*(default=3600, minimum=0, maximum=65535)*

Defines the number of seconds between automatic routing broadcasts issued by the node. The default value of 3600 specifies an hourly broadcast. In addition, broadcasts can be disabled altogether by setting this parameter to zero.

---

## 8. Network "time-to-live" initializer

---

*(default=64, minimum=0, maximum=255)*

Defines the initial value of the "time-to-live" field in the Network Header of all network-layer frames originated by this node. The time-to-live field is decremented by each intermediate node that relays the frame. If the time-to-live value ever reaches zero, the frame is discarded. This protects the network against frames persisting forever as the result of a routing loop. The value of this parameter should be a bit larger than number of "hops" in the longest legitimate route in the network.

---

## 9. Transport timeout (seconds)

---

*(default=60, minimum=5, maximum=600)*

Defines the number of seconds between transport-layer retries.

---

## 10. Transport maximum tries

---

*(default=3, minimum=2, maximum=127)*

Defines the maximum number of transport-layer tries attempted before a circuit failure is reported.

---

## 11. Transport acknowledge delay (seconds)

---

*(default=3, minimum=1, maximum=60)*

Defines the number of seconds' delay used by the transport layer from the time it receives an information message until it sends an information-acknowledge message. The purpose of this delay is to give the acknowledgement an opportunity to be "piggybacked" upon an outgoing information message.

---

## 12. Transport busy delay (seconds)

---

*(default=180, minimum=1, maximum=1000)*

Defines the maximum number of seconds that the transport layer will remain "choked" as the result of an incoming message that has the choke flag bit set. The purpose of this timeout is to prevent an indefinite hangup in the event that the "unchoke" message is lost.

---

## 13. Transport requested window size (frames)

---

*(default=4, minimum=1, maximum=127)*

Defines the maximum number of incoming out-of-sequence information messages that the transport layer will buffer while waiting for the next expected information message to arrive. Also defines the maximum number of outgoing information messages that the transport layer will send without receiving acknowledgement.

---

## 14. Congestion control threshold (frames)

---

*(default=4, minimum=1, maximum=127)*

Defines the maximum allowable backlog of messages that the transport layer will buffer before it sends a choke message. Also defines the maximum allowable backlog of frames that the link layer will buffer before it sends an RNR control frame.

---

## 15. No-activity timeout (seconds)

---

*(default=900, minimum=0, maximum=65535)*

Defines the maximum number of seconds that a transport-layer circuit or a link-layer connection can remain idle (i.e., no information transfer in either direction) before it is automatically disconnected. Also defines the maximum time that a CQ request can remain active.

## 16. P-persistence ( $p=P/256$ )

---

*(default=64, minimum=0, maximum=255)*

Together with slot time (parameter #17), defines the exponential delay algorithm used by the node when keying up its transmitter. When the node has something to transmit and the channel is clear, the node generates a random integer in the range 0–255. If the random number is *less than or equal to* the P-persistence parameter, the node keys up its transmitter immediately. Otherwise, the node delays for one slot time, generates a new random number, and repeats the procedure. The default value of 64 corresponds to a probability of 0.25.

## 17. Slot time (10 ms increments)

---

*(default=10, minimum=0, maximum=127)*

Together with P-persistence (parameter #16), defines the exponential delay algorithm used by the node when keying up its transmitter. The default value of 10 corresponds to a slot time of 100 milliseconds.

## 18. Link T1 timeout “FRACK” (seconds)

---

*(default=4, minimum=1, maximum=15)*

Defines the number of seconds between link-layer retries. When digipeating is used, this value is multiplied by  $2D+1$ , where D is the number of digipeaters.

## 19. Link xmit window “MAXFRAME” (frames)

---

*(default=7, minimum=1, maximum=7)*

Defines the maximum number of outgoing information frames that the link layer will send without receiving acknowledgement.

## 20. Link maximum tries

---

*(default=10, minimum=0, maximum=127)*

Defines the maximum number of tries that the link layer will attempt before reporting a link failure. If this parameter is set to zero, the link layer will retry forever (*not recommended*).

## 21. Link T2 timeout (10 ms increments)

---

*(default=100, minimum=0, maximum=6000)*

Defines the delay (measured in 10-millisecond increments) used by the link layer from the time it receives an information frame until it sends an acknowledgement (RR, RNR, or REJ) control frame. The purpose of this delay is to give the acknowledgement an opportunity to be “piggybacked” upon an outgoing information frame.

---

## 22. Link T3 timeout (10 ms increments)

---

*(default=18000, minimum=0, maximum=65535)*

Defines the maximum no-activity period (measured in 10-millisecond increments) permitted by the link layer before it issues a poll to make sure the link is still intact. This timeout is also used to break link-layer choke deadlocks.

---

## 23. AX.25 digipeating? (1=on, 0=off)

---

*(default=1, minimum=0, maximum=1)*

Defines whether or not the node will perform AX.25 digipeating. The default value of 1 causes digipeating to be enabled.

---

## 24. Callsign validation? (1=on, 0=off)

---

*(default=1, minimum=0, maximum=1)*

Defines whether or not the node will perform validation checks on amateur callsigns. The default value of 1 causes callsign validation to be enabled. (This parameter is ignored in commercial version of NET/ROM.)

---

## 25. Station ID? (2=on, 1=conditional, 0=off)

---

*(default=2, minimum=0, maximum=2)*

Defines whether the node broadcasts station-identification beacons. The default value of 2 causes station identification to be broadcast every 10 minutes. The value 1 causes station identification to be broadcast only if the node has transmitted since the last beacon. The value 0 disables station identification beacons altogether.

---

## 26. CQ broadcasts? (1=on, 0=off)

---

*(default=1, minimum=0, maximum=1)*

Defines whether or not the node will broadcast AX.25 UI-frames in response to the CQ command. Even if such broadcasts are disabled by setting this parameter to zero, the other features of the CQ command continue to operate normally. The default value of 1 causes CQ broadcasts to be enabled.



# Index

---

## A

- Abbreviating commands, 9
- Aborting a CONNECT command, 10, 52
- Aborting a CQ command, 52
- Adaptive routing, 45
- Adopting a user's callsign, 34
- Alias, 9, 48
- Alternate routes, 1, 4, 45
- Amateur callsigns, validation of, 57
- Analogy to telephone system, 33
- Answering a CQ, 8, 11
- Assigning path quality values, 55
- Automatic routing, 45
- Automatic routing updates, 4, 46, 57, 65
- Automatic routing broadcasts, 46
- Avoiding locked routing entries, 56
- AX.25 link-layer protocol, 1, 2, 59

## B

- Backbone, 36
- Baud rates
  - host B-command, 22
  - host ESC-B command, 19
  - TNC-2 DIP switches, 29
- Blocks, 21
- Broadcasts
  - CQ, 10
  - routing, 46, 65
  - station identification, 48
- Bulletin board systems, 1
- "Busy from..." message, 6, 7

## C

- Calculations, route quality, 66
- Callsigns
  - hard-coded into EPROM, 30
  - setting or verifying, 30
  - validating amateur, 57
- Channel quality, 47
- Checking if you have a decent path, 51
- Circuits, 1, 9, 18, 35, 40
- Circuit manager, 40, 43
- Command blocks, 22
- Command interpreter, 41-44
- Commands, 4, 9
- Commands
  - CONNECT, 4-7, 9, 34, 40, 41, 44, 45, 52
  - CQ, 1, 3, 4, 8, 10, 11, 41, 52
  - IDENT, 4, 12, 17, 41, 49
  - NODES, 4, 6, 12, 13, 17, 40, 41, 45, 46, 49, 51, 63, 66
  - PARMS, 4, 14, 17, 41, 49, 67-71
  - RESET, 4, 15, 17, 41, 49
  - ROUTES, 4, 16, 17, 40, 41, 49, 55, 63
  - SYSOP, 4, 12, 24, 15, 17, 20, 41, 46, 49
  - USERS, 4, 8, 11, 18, 41
- Computer interface, 21
- CONNECT, 4-7, 9, 34, 40, 41, 44, 45, 52
- Connect host
  - C-command, 22
  - ESC-C command, 20
- "Connected to..." message, 5, 7
- Connections
  - making, 33
  - to distant station, 6
  - to host, 7, 10
  - to nearby station, 5
- Control operator, 17, 49
- Control operators, notes for, 54
- Coupler, diode-matrix, 1, 3, 30, 32
- CPU clock speed, 28
- CQ, 1, 3, 4, 8, 10, 11, 41, 52
- Cross-frequency operation, 1
- Crosslink, 7, 36, 43
- Crosslink, digipeated, 56

## D

- Data switch, 41-44
- Datagram network service, 1
- DCD-B wired to RS232 port, 28
- Deferred disconnect, 3, 48
- Destination list, 12, 45, 63
- Diagnostic messages, 6, 7
- Digipeated crosslink, 56
- Digipeating, 1, 2, 4, 9, 37, 49
- Digipeating vs. store-and-forward, 37
- Diode-matrix coupler, 3, 30
- Disabling automatic routing updates, 64
- Disconnect host
  - D-command, 23
  - ESC-D command, 20
- Disconnecting, 6
- Distant station, connecting to, 6
- Don't backtrack, 52
- Don't repeat commands, 52
- Downlink, 5, 9, 18, 34, 44
- Dual-channel node, 1, 3, 32, 36
- Duplex repeater, 58

## E

- End-to-end transport protocol, 60
- EPROM, 2, 30
- Error control, 2, 40, 48
- Error messages, 6, 7
- Error recovery, 38
- ESC-B host baud-rate, 19
- ESC-C, host connect, 20
- ESC-D, host disconnect, 20
- ESC-P, host password, 20
- ESC-T, host TXDELAY, 21
- ESC-Y, host incoming connections, 21

## F

- "Failure with..." message, 6, 7
- Firmware organization, 39
- Fixed limits, 54
- Flow control, 2, 40, 48
- Flow of information, 42

## G

- Get a message, host G-command, 23
- Guided tour, 5

## H

- Hardware setup
  - PK-87/PK-90, 30
  - TNC-2 and "clones", 27
- HDLC driver, 39
- HDLC frames, structure of, 59
- Hidden nodes, 13
- Host commands (computer interface)
  - B-command, baud rate, 22
  - C-command, connect, 22
  - D-command, disconnect, 23
  - G-command, get a message, 23
  - I-command, identity, 24
  - M-command, monitor, 24
  - P-command, password, 24
  - Q-command, resynchronize, 25
  - S-command, send a message, 25
  - T-command, TXDELAY, 26
  - Y-command, incoming connections, 26
- Host commands (terminal interface)
  - ESC-B baud-rate, 19
  - ESC-C, connect, 20
  - ESC-D, disconnect, 20
  - ESC-I, identity, 20
  - ESC-P, password, 20
  - ESC-T, TXDELAY, 21
  - ESC-Y, incoming connections, 21
- Host, connecting to, 7, 10
- Host interface, 19-26, 41
- Host NET/ROM, 1, 19
- Host terminal cable, 29
- HSTX, HETX, HDLE, 21

## I

- IDENT, 4, 12, 17, 41, 49
- Identifier, mnemonic, 3, 9, 12, 31
- Identity, host interface
  - ESC-I command, 20
  - I-command, 24
- Incoming connections, host interface
  - ESC-Y command, 21
  - Y-command, 26
- Information flow, 42
- Installing NET/ROM, 27-30
- Inter-node HDLC frames, 59
- Interconnect, TNC-to-TNC, 3
- Interconnect cable, 32
- Interconnect protocol, 62
- "Invalid command" message, 8, 11, 16
- ISO reference model, 1, 39

## L

Link, 34  
Link manager, 39, 42-44  
Local routing table updates, 13, 46  
Locked  
    neighbor-list entries, 16  
    routing entries, 13, 56, 66  
Locking out an unuseable path, 16, 55  
Long frames, 53

## M

Manual routing updates, 4, 13  
Maximum fixed limits, 54  
Messages, diagnostic, 6, 7  
Mnemonic node identifier, 3, 9, 12, 31  
Monitor, host M-command, 24  
Multi-channel node, 1, 3, 4, 30, 32, 36  
Multi-port digipeating, 49  
Multiple uplinks, 53

## N

Nearby station, connecting to, 5  
Neighbor list, 16, 45, 63  
Networking, 1  
Node, 1, 9  
Node notes, 51  
NODES, 4, 6, 12, 13, 17, 40, 41, 45, 46, 49,  
    51, 63, 66  
Notes  
    for control operators, 54  
    for users, 51

## O

Obsolescence count, 13, 46, 51, 64, 66  
Op-amp, upgrade U3 in TNC-2, 29  
Organization of the firmware, 39

## P

Parameters, 14, 57, 67-71  
PARMS, 4, 14, 17, 41, 49, 67-71  
Password string, 17, 20, 24, 31, 41, 49  
Password, setting  
    ESC-P command, 20  
    P-command, 24  
Patchcord, 18, 41, 43  
Path quality, 16, 47, 55  
PK-87/PK-90 TNCs, 1, 2, 19, 30  
PK-87/PK-90 hardware setup, 30  
Protocol, 1, 59  
Protocol delimiters HSTX/HETX/HDLE, 21  
Protocol identification (PID), 40  
PTT failsafe timer, 28

## Q

Quality  
    calculations, 66  
    channel, 47  
    path, 47  
    route, 13, 45-47, 51

## R

Radio problems, 54  
RAM  
    free space displayed by USERS, 18  
    upgrade TNC-2 to 32K, 27  
Remote updates, 13, 46  
RESET, 4, 15, 17, 41, 49  
Response blocks, 22  
Resynchronize, host Q-command, 25  
Route quality, 13, 45-47, 51, 64  
Route quality calculations, 66  
ROUTES, 4, 16, 17, 40, 41, 49, 55, 63  
Routing, 1, 45, 63  
Routing  
    algorithm, 64  
    automatic, 4  
    broadcast, 46, 65  
    entries, locked, 66  
    manager, 40, 43, 45, 64  
    table, 12, 16, 43, 45, 63  
    updates, 4, 46, 65  
RS232  
    driver, 39  
    interconnect protocol, 62

## S

- Satellite links, 3
- Security, 49
- Send a message, host S-command, 25
- Sending very long frames, 53
- Serial drivers, 39
- Sliding window protocol, 2, 40, 48
- Software setup, 30
- SSID suffixes, 49
- SSID, mapping N to 15-N, 35
- Station identification, 48, 54
- Store-and-forward, 1, 2, 38
- SYSOP, 4, 12, 24, 15, 17, 20, 41, 46, 49

## T

- Telephone links, 3
- Telephone system, analogy to, 33
- Terminal, host interface, 19
- Timer, PTT failsafe, 28
- TNC-2 and "clones", 1, 2, 27
- Transport layer
  - service, 1
  - circuit, 35
  - protocol, 60
- TXDELAY, setting
  - ESC-T command, 21
  - T-command, 26

## U

- Updates, routing table, 4
- Unattended operation, 20, 21, 31
- Unproto mode CQ broadcasts, 10
- Unusable paths, 55
- Upgrade RAM to 32K, 27
- Uplink, 5, 6, 18, 34, 42
- Uplinks, multiple, 53
- Use count, 16
- USERS, 4, 8, 11, 18, 41
- Users, notes for, 51

## V

- Validation
  - amateur callsigns, 57
  - control operator, 17, 49
- Very long frames, 53
- Virtual circuit, 1, 35

## W

- Warm-start reset, 15
- Watchdog timer, 54
- Wormhole, 3



